Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Simulation à long terme d'un réseau WAN de données cas du LHC, CERN

Gilbert, Thomas

Award date:
1997

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Simulation à long terme d'un réseau WAN de données

- Cas du LHC, CERN -

Thomas GILBERT

Mémoire réalisé par Thomas GILBERT

en vue de l'obtention du diplôme de « Maître en Informatique »,

suite à son stage en entreprise au CERN (site de Meyrin), Genève, Suisse
du 1er août 1996 au 31 janvier 1997.

Promoteur : Monsieur Philippe van Bastelaer

Maître de stage : Monsieur Julian BUNN

Résumé :
Ce mémoire a pour objectif de présenter l'évolution d'un simulateur de réseau, en partant d'une version « prototype » développée au CERN jusqu'à la version OO implémentée lors de mon stage. Quelques pistes sont également lancées afin de donner une chance au simulateur d'aller plus loin dans la modélisation d'un réseau WAN.

Mots-clés : architecture OO, simulation, générateur de nombres aléatoires, réseau WAN.

Table des matières

Avant-Propos

Je tiens à remercier sincèrement tous ceux qui m'ont assisté lors de la réalisation de ce travail ou durant mon stage au CERN. Je dédie tout particulièrement ces remerciements à Julian Bunn, François Fluckiger et Bryan Carpenter à qui je dois une multitude de précieux conseils ainsi qu'à Philippe van Bastelaer qui supervisa de manière efficace la rédaction de ce mémoire.

1. Introduction : le projet LHC

Ce projet de simulation de réseaux s'inscrit dans un projet plus global, de très grande envergure, dans lequel beaucoup de pays et d'organisations scientifiques du monde entier ont investi à la fois des ressources humaines de taille et des moyens financiers considérables. Ainsi, de nombreux outils ont été mis au point pour aider les décideurs à choisir les solutions qui répondent le mieux aux besoins et aux moyens de chacun...

1.1 Le CERN

Durant la première moitié du siècle, les découvertes en physique ne manquent pas, de la mise en évidence des électrons au noyau atomique et ses constituants, de la relativité à la mécanique quantique. Malheureusement, les conflits des années 30 et 40 ont interrompu cette vague de progrès, les scientifiques étant contraints de partir vers des pays plus calmes. Le retour de la paix favorisa enfin le retour des découvertes mais, au début des années 50, les Américains comprirent que des outils sophistiqués devenaient indispensables pour permettre au progrès de continuer et que des investissements dans le domaine de la science fondamentale pouvaient stimuler le développement économique et technologique. Alors que les scientifiques d'Europe se fiaient toujours à des équipements simples basés sur la radioactivité et les rayons cosmiques, de puissants accélérateurs furent construits aux Etats-Unis.

Une poignée de scientifiques perçut enfin que la coopération était le seul chemin pour aller de l'avant dans la recherche en Europe. Malgré les bonnes traditions intellectuelles et les prestigieuses universités, aucun pays européen ne pouvait se débrouiller seul. La création d'un laboratoire européen fut même recommandée lors d'une réunion de l'UNESCO à Florence en 1950 et, pas moins de trois ans plus tard, une convention fut signée par les pays du conseil européen pour la recherche nucléaire. Le CERN était né et, avec lui, le prototype d'une chaîne d'institutions européennes compétentes dans le domaine de l'espace, de l'astronomie et de la biologie moléculaire. Ainsi, l'Europe était en position de regagner une place de choix dans le monde de la science.

Le CERN existe principalement pour munir les physiciens européens d'un accélérateur de particules qui rencontre les exigences de la recherche aux limites de la connaissance humaine. Les premiers accélérateurs importants [CRN97-1] furent l'ISR (Intersecting Storage Ring, 1971) et le SPS (Super Proton Synchrotron, 1981) qui permit la production de grandes quantités de particules W et Z deux ans plus tard, confirmant ainsi les théories des forces électromagnétiques et des forces faibles. Actuellement, c'est le LEP (Large Electron-Positron collider), implanté dans un tunnel circulaire de 27 km sur la frontière franco-suisse, qui permet plus de mesures et une meilleure qualité de collisions. La puissance du LEP devrait être doublée pour ouvrir la porte à de nouvelles découvertes. Ainsi, des résultats plus précis et plus abondants sont attendus durant le reste de cette décennie, ce qui devrait améliorer sensiblement notre compréhension de la matière.

1.2 Le LHC

Les scientifiques disposent actuellement de la preuve que la réponse à quelques-unes des questions les plus profondes de notre temps réside au-delà de la barrière du TeV[1]. Le LHC (Large Hadron Collider) est un accélérateur de particules qui devrait amener les protons à entrer en collision à de plus grands niveaux d'énergie (14 TeV) encore jamais atteints. Ceci devrait permettre aux scientifiques d'explorer encore plus loin les structures de la matière et de recréer les conditions qui étaient celles de l'univers juste 10^{-12} secondes après le « Big Bang » alors que la température était de 10^{16} degrés. Pour cela, des systèmes de refroidissement considérables seront nécessaires, soit un total de 12 millions de litres d'azote liquide et de 700 000 litres d'hélium liquide [CRN97-2].

Le LHC est conçu pour partager le tunnel du LEP et sera alimenté par les sources existantes de particules. Cette nouvelle machine, un véritable challenge, utilisera les aimants superconducteurs les plus avancés ainsi que d'autres technologies des accélérateurs encore jamais utilisées pour permettre d'observer des phénomènes qui ne sont encore que des prédictions théoriques.

1.3 Le réseau de données

L'étendue du projet et le nombre d'institutions internationales ayant un intérêt direct ou indirect à consulter les résultats ou à participer aux expériences ont rendue indispensable la mise en place d'un réseau informatique qui interconnecte les organismes concernés. Grâce à ce réseau, les différents membres pourront consulter les données et les résultats des différentes expériences, visualiser les photos de collisions de particules ou encore participer au décodage et au dépouillement des chiffres fournis par les nombreux instruments de mesure qui jalonneront le LHC.

1.4 Les configurations proposées

La structure du réseau et la quantité de ressources à rendre disponibles dépendent fortement du rôle joué par chacun dans l'ensemble des organisations connectées.

---

[1] TeV : Tera électronvolt (10^{12} électronvolts). L'électronvolt mesure l'énergie cinétique acquise par un électron accéléré sous une différence de potentiel de 1 volt. (Valeur approximative d'un eV : $1{,}6.\ 10^{-19}$ joule)

Ainsi, trois configurations de base peuvent être mises en évidence :

- Modèle ce(r)ntralisé.
- Modèle distribué
- Modèle des centres régionaux

1.4.1 Le modèle centralisé

La structure centralisée du réseau de données considère que tout le traitement des nombres fournis par les instruments de mesure est pris en charge par le CERN qui les conserve sur les disques de son réseau local. Les résultats de ces traitements de données sont quant à eux disponibles pour toutes les autres institutions du réseau qui pourront les consulter librement. Le Schéma 1 ci-dessous illustre cette configuration en supposant un total de 7 institutions. Il va sans dire que cette configuration est vraisemblablement la plus lourde et la plus coûteuse pour le CERN qui prend en charge la totalité du traitement.

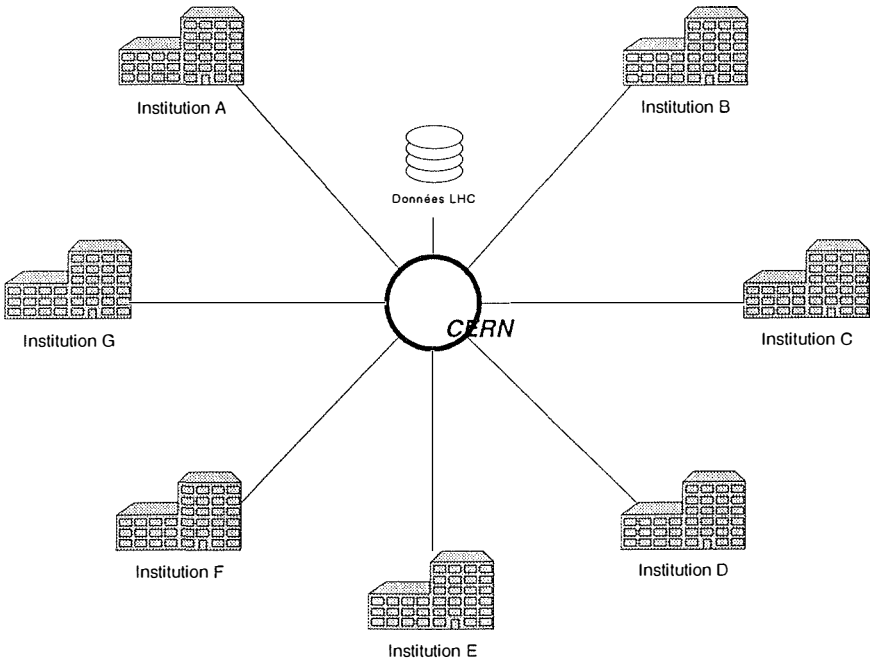


Schéma 1 : Le modèle centralisé

1.4.2 Le modèle distribué

A l'inverse, le modèle distribué suppose un partage du traitement des nombres donnés par les instruments de mesure parmi toutes les institutions connectées, comme illustré au Schéma 2. Chaque organisme se voit donc confier une partie de la tâche de traitement et est supposé mettre les résultats obtenus à la disposition de tous (y compris le CERN qui en recevra une copie). Cette configuration permet de répartir la charge (en ressources et en argent) inhérente au traitement des données mais elle suppose une participation de chaque institution, ce qui est difficile à imaginer.

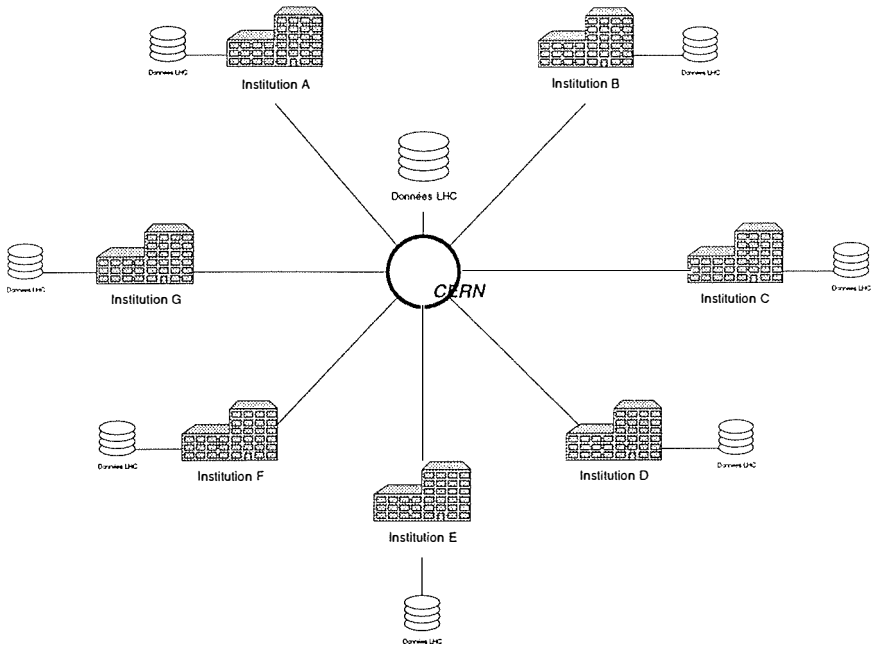


Schéma 2 : Le modèle distribué

1.4.3 Le modèle des centres régionaux

Le modèle des centres régionaux est un modèle intermédiaire entre le modèle centralisé et le modèle distribué. En effet, comme le montre le Schéma 3 ci-après, les informations à traiter sont distribuées parmi un petit nombre d'organisations (les institutions A, B et C dans l'exemple fictif proposé) généralement proches du CERN, géographiquement parlant. Comme dans le modèle distribué, ces institutions seront chargées de prendre en charge le traitement d'une partie des données fournies par le CERN et de lui renvoyer une copie des résultats.

Les autres institutions peuvent se connecter au CERN ou aux organismes faisant partie de la configuration distribuée afin de consulter les résultats des traitements.

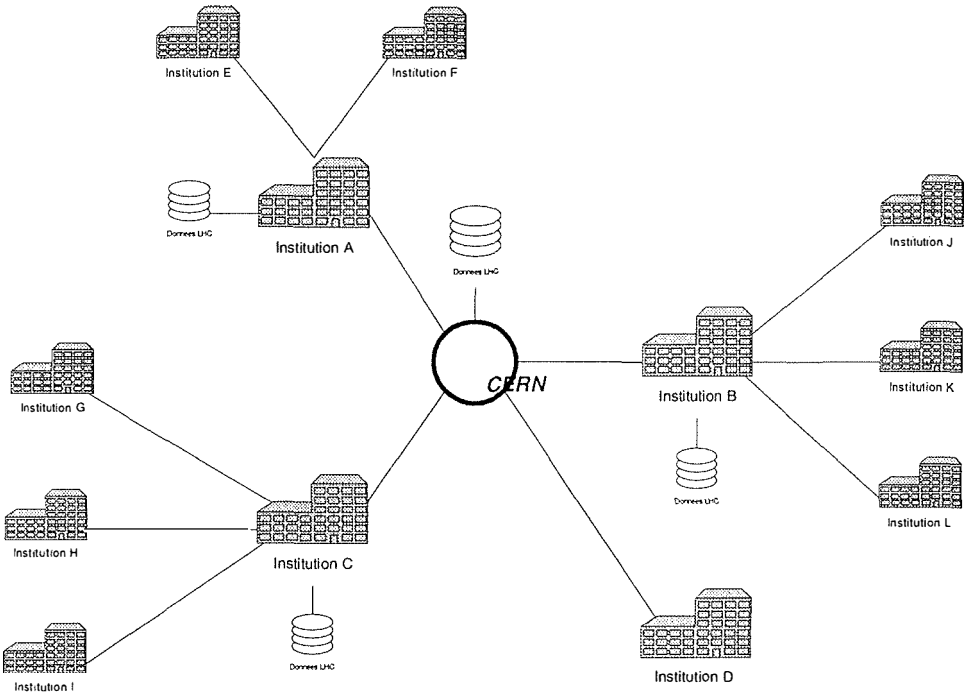


Schéma 3 : le modèle des centres régionaux

1.5 Le rôle des simulateurs

1.5.1 Les ressources clés

Les trois modèles exposés ci-dessus possèdent chacun leurs propres exigences en terme de ressources :

- Les ressources CPU : le traitement des données fournies par les instruments de mesure du CERN nécessite un certain nombre de processeurs.
- Les ressources « Espace disque » : les données à traiter ainsi que le résultat des traitements doivent être stockés sur disque.
- Les ressources « Largeur de bande » : les nombreux transferts de données avec les autres institutions exigent une certaine largeur de bande.
- Les ressources financières : les processeurs, l'espace disque mais aussi l'utilisation des lignes ont un prix !

Vu le nombre de transferts, la taille des données à traiter et le nombre d'institutions, il est difficile de dire quelle est la configuration la plus acceptable pour chacun. Ainsi, le développement de simulateurs capables de permettre une évaluation des différentes ressources nécessaires pour chaque modèle a été proposé au CERN.

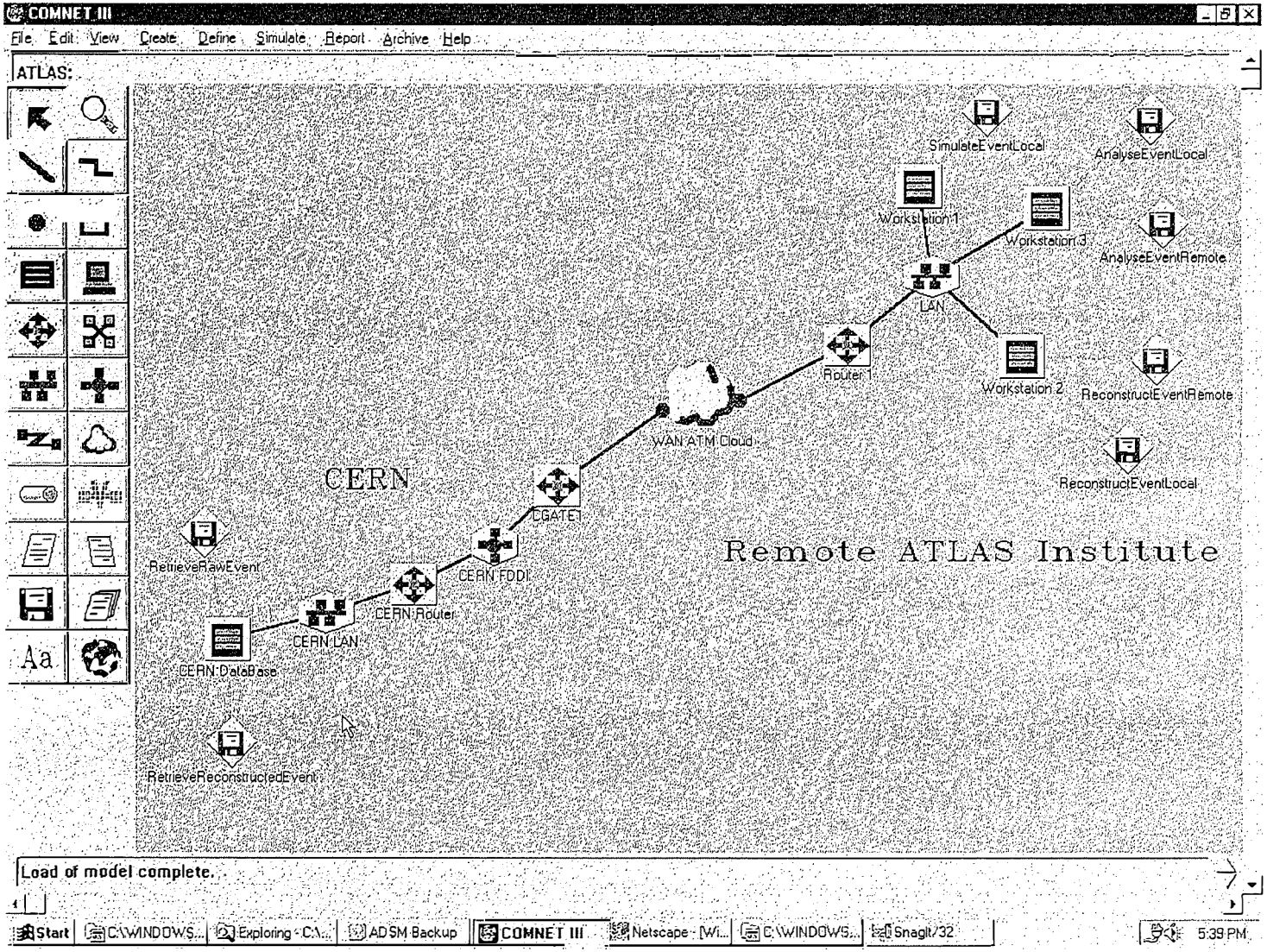
1.5.2 Les simulateurs commerciaux

Avant de lancer un projet de développement d'un simulateur « à la carte », il paraissait important de consulter le marché. Il existe en effet de puissants simulateurs qui permettent de modéliser une configuration de réseau, d'en définir les nombreux paramètres et d'en obtenir une quantité appréciable d'informations importantes comme le trafic, la charge des noeuds, les taux d'erreur, etc.

Parmi les outils disponibles sur le marché, le CERN a retenu le logiciel « COMNET III ». Après avoir acquis une version d'évaluation du simulateur, il est apparu combien ce logiciel était puissant et... compliqué. Il nécessite en effet de modéliser le réseau de manière extrêmement précise et complète : il faut spécifier le type et les paramètres de chaque réseau, routeur, ligne, base de données, etc. Outre le fait qu'ils rendent la modélisation très ardue, tous ces paramètres sont difficiles à identifier. En effet, la mise en place du réseau de données ne devrait se faire que dans une dizaine d'années et nul ne peut encore prévoir aussi précisément quel sera le niveau de la technologie et quel sera exactement le matériel utilisé.

Parallèlement, la taille du réseau (qui comptera certainement plusieurs dizaines d'institutions) nécessitera un modèle de très grande taille et peu lisible, si l'on en croit la copie d'écran ci-dessous (Ecran 1) du logiciel sur lequel ne figure qu'une seule institution (plus le CERN).

Bref, COMNET III, bien que très puissant, est plutôt destiné à la modélisation de réseaux contemporains dont on connaît précisément la structure et les composants. D'autres outils de ce genre (comme OPNET ou NETWORK II) sont disponibles mais « souffrent » eux aussi d'une trop grande complexité par rapport au problème posé.



Ecran 1 : Exemple de réseau simple modélisé avec COMNET III

1.5.3 Un simulateur développé au CERN

La décision fut donc prise de développer un simulateur au CERN qui permette de modéliser à long terme une configuration plus générale de réseau dont on ignore les niveaux les plus bas.

Le simulateur devra donner à l'utilisateur la possibilité de spécifier une configuration de réseau (par exemple, un des trois modèles ci-avant), de définir où et quand se font les traitements et les transferts de données et sera capable de simuler le comportement du réseau modélisé afin de fournir une évaluation grossière des ressources nécessaires et des coûts pour chacun.

L'objectif de ce simulateur n'est pas de fournir des chiffres exacts, proches de ceux de la réalité car le principal souci des utilisateurs est plutôt de pouvoir comparer différentes structures de réseau, de tester si les différences de coût sont grandes ou encore d'avoir un avis sur la faisabilité d'une configuration.

Bref, les résultats donnés par le simulateur développé ne devraient pas avoir un poids décisif dans la prise de décision quant à la structure du futur réseau de données. Ils seront plutôt un avis de support voire de validation d'une solution envisagée.

Ainsi, le chapitre 2 présentera la première version d'un simulateur développée au CERN par Julian BUNN au début de l'année 1996. Le chapitre 3 tente, quant à lui, de vérifier que les résultats de ce simulateur ne sont pas entachés d'erreurs dues à un mauvais générateur de nombres aléatoires. L'objectif du chapitre 4 est de proposer une nouvelle version orientée objet du simulateur, Modnet, tandis que le chapitre 5 tente de lancer quelques pistes d'améliorations qui pourraient être apportées à cette dernière implémentation.

2. *Une première version du simulateur*

Début 1996, une première version du simulateur fut écrite au CERN par Julian Bunn. Malgré un modèle relativement simple et restrictif, les résultats obtenus soulevèrent un grand intérêt parmi la communauté scientifique du CERN qui encouragea, par la suite, la poursuite du projet.

2.1 Concepts de base

Le but du programme est donc de simuler l'évolution de l'exécution de diverses tâches au sein d'instituts répartis dans le monde entier et reliés par un niveau informatique.

2.1.1 Les instituts

Ces *instituts* regroupent le CERN, des universités, des centres de recherche ou toute autre organisation qui participe au projet LHC ou qui pourrait trouver un quelconque intérêt dans les résultats fournis.

On suppose que ces instituts font tous partie d'un réseau commun, qu'il soit privé ou de type *internet*. La largeur de bande du réseau local des instituts ainsi que celle des lignes qui les relient sont supposées être fixes et identiques pour tous les instituts.

2.1.2 Les tâches

Chaque institut générera un certain nombre de *tâches* à une fréquence moyenne donnée. Chaque tâche consiste en une série d'étapes simples pouvant être d'un des types suivant :

- Envoi de données vers un autre institut déterminé
- Envoi de données vers un autre institut choisi au hasard
- Envoi de données vers le réseau local
- Demande d'envoi de données d'un institut déterminé
- Demande d'envoi de données d'un institut choisi au hasard
- Demande d'envoi de données du réseau local
- Traitement de données

Chaque étape est mesurée en octets (excepté le dernier type qui est mesuré en secondes sur un processeur de 100 MIPS) et ne peut commencer que si l'éventuelle étape qui la précède est terminée.

2.1.3 Les résultats de la simulation

Le but de la simulation est de générer aléatoirement les tâches au sein de chaque institut, conformément aux fréquences données, de simuler leur évolution ainsi que les éventuels transferts de données et de donner quelques observations sur l'état du système, à savoir :

- ✓ *Le nombre de tâches qui utilisent chaque ligne* ; ce qui peut être interprété comme, par exemple, le nombre de canaux virtuels au sens ATM ou de circuits virtuels au sens X25. Une tâche n'est considérée comme « utilisant une ligne » que si elle a déjà entamé un transfert de données et qu'elle ne l'a pas encore terminé.
- ✓ *Le nombre de processeurs utilisés*, en sachant qu'il n'y a aucune « concurrence » dans l'allocation des ressources d'un processeur ; celui-ci n'est donc pas partagé et ne donne la main à une autre tâche que si la précédente est terminée.
- ✓ *Le nombre de tâches en cours d'exécution* au sein de chaque institut, c'est-à-dire le nombre de tâches qui y sont exécutées, qu'elles soient en cours de transfert ou de traitement de données.
- ✓ *Le taux de transfert moyen par tâche* calculé en effectuant le rapport entre la quantité de données transférées et le temps moyen consacré aux transferts.
- ✓ *L'utilisation des disques* au sein de chaque institut, en sachant que les données reçues et les données envoyées sont stockées.
- ✓ *Le montant des coûts* pour chaque institut. Ce montant couvre le nombre de processeurs requis, l'espace disque nécessaire au stockage des données transférées ainsi que la location des lignes entre instituts. Il est calculé en se basant sur les coûts unitaires donnés par l'utilisateur.

La quasi-totalité de ces informations n'est affichée qu'en fin de simulation ; les autres (essentiellement le nombre de tâches en cours d'exécution et le nombre de tâches utilisant une ligne) sont affichées lors de la simulation, sous forme agrégée à tous les instituts, selon une fréquence spécifiée par l'utilisateur.

Ce sont principalement le montant des coûts et les ressources matérielles minimales dont il faut disposer (processeurs, espace disque et largeur de bande) qui intéressent les différents décideurs. Les autres informations sont surtout utiles à celui qui utilise le simulateur et qui tente de trouver un modèle pertinent.

2.2 Principes & fonctionnement

2.2.1 Structure du programme et langage utilisé

Cette version du simulateur était implémentée en Fortran 77 et ne profitait donc pas des structures orientées objets, des pointeurs ou autres outils perfectionnés. Ainsi, la « base de données » du logiciel consiste simplement en un ensemble de tableaux dont la taille est statique et fixée par quelques constantes définies au début du listing.

La structure du programme ne comporte qu'un seul bloc. En effet, on n'observe aucune procédure ou fonction auxiliaire, seulement quelques *goto* utilisés pour les boucles.

Les quatre grandes étapes de programmes sont relativement classiques et se résument ainsi :

- ✓ Initialisation des tableaux et des variables globales
- ✓ Lecture des données dans un fichier « model.dat »
- ✓ Simulation avec impression régulière des résultats intermédiaires à l'écran ou dans un fichier
- ✓ Impression des résultats finaux à l'écran ou dans un fichier.

2.2.2 Structures de données

Comme mentionné au point 2.2.1, les principales structures de données sont organisées sous forme de tableaux statiques. Ceux-ci rassemblent les informations relatives :

- aux instituts
- aux tâches générées par les instituts
- au réseau qui lie les instituts entre eux.

Le Schéma 4, commenté en détail ci-dessous, montre comment les données sont intuitivement structurées. Dans cette section, certaines notations seront utilisées pour faciliter l'expressivité de la sémantique de chaque élément. Ainsi, les mots **en gras** désigneront des entités ou des associations, les mots en *italique* représenteront

des noms de rôle tandis que les mots en `caractère de type courrier` référenceront un attribut d'une entité.

☞ *Lecture du schéma*

Institut représente le concept d'institut tel qu'il a été introduit au point 2.1.1. Identifié par un nom, il possède de nombreux attributs dont les principaux sont décrits dans le Tableau 1 ci-dessous. Chaque **institut** *est connecté* à au moins un autre **institut** grâce à une ligne de capacité `C-LargBande`[2].

Tableau 1: Les attributs de l'entité "Institut"

Nom de l'attribut	Commentaires
Nom	Nom de l'institut
I-ResLocal	Largeur de bande du réseau local (Mbps)
I-CoûtConnect	Coût de location d'une ligne de 2 Mbps pendant un mois ($)
I-CoûtProc	Coût d'acquisition d'un processeur 100 MIPS ($)
I-CoûtDisque	Coût d'acquisition d'un espace disque de 100 GB ($)
I-nTâches	Nombre de tâches en cours d'exécution
I-nTraitement	Nombre de tâches en phase de traitement de données
I-ProcMax	Nombre maximum de processeurs requis depuis le début de la simulation
I-DisqueUtil	Espace disque utilisé depuis le début de la simulation

Chaque **institut** *planifie* éventuellement un ou plusieurs **type(s) de tâche** en spécifiant certaines propriétés de **planification** décrites dans le Tableau 2. Chaque **type de tâche** planifié, identifié par son nom `T-Nom`, *est composé d'*une ou plusieurs **étapes** dont chacune, selon son type `E-Type`, pourra *demander un transfert avec* un **institut**. Les attributs de l'entité **étape** sont brièvement décrits dans le Tableau 3.

Lors de la simulation, conformément à la **planification**, les **instituts** *génèrent* un certain nombre de **tâches** qui *exécutent* successivement les différentes **étapes** qui *composent* le **type de tâche** correspondant. Pour ce faire, ces **tâches** peuvent *utiliser* un **canal virtuel** grâce auquel un **institut** *envoie des données* à un autre **institut** qui *reçoit ces données.*

[2] Il n'y a pas de différence de sémantique entre le rôle « Est relié à » et le rôle « est connecté à » ; le choix des termes est uniquement dû au caractère unique du nom des rôles d'un schéma ERA.

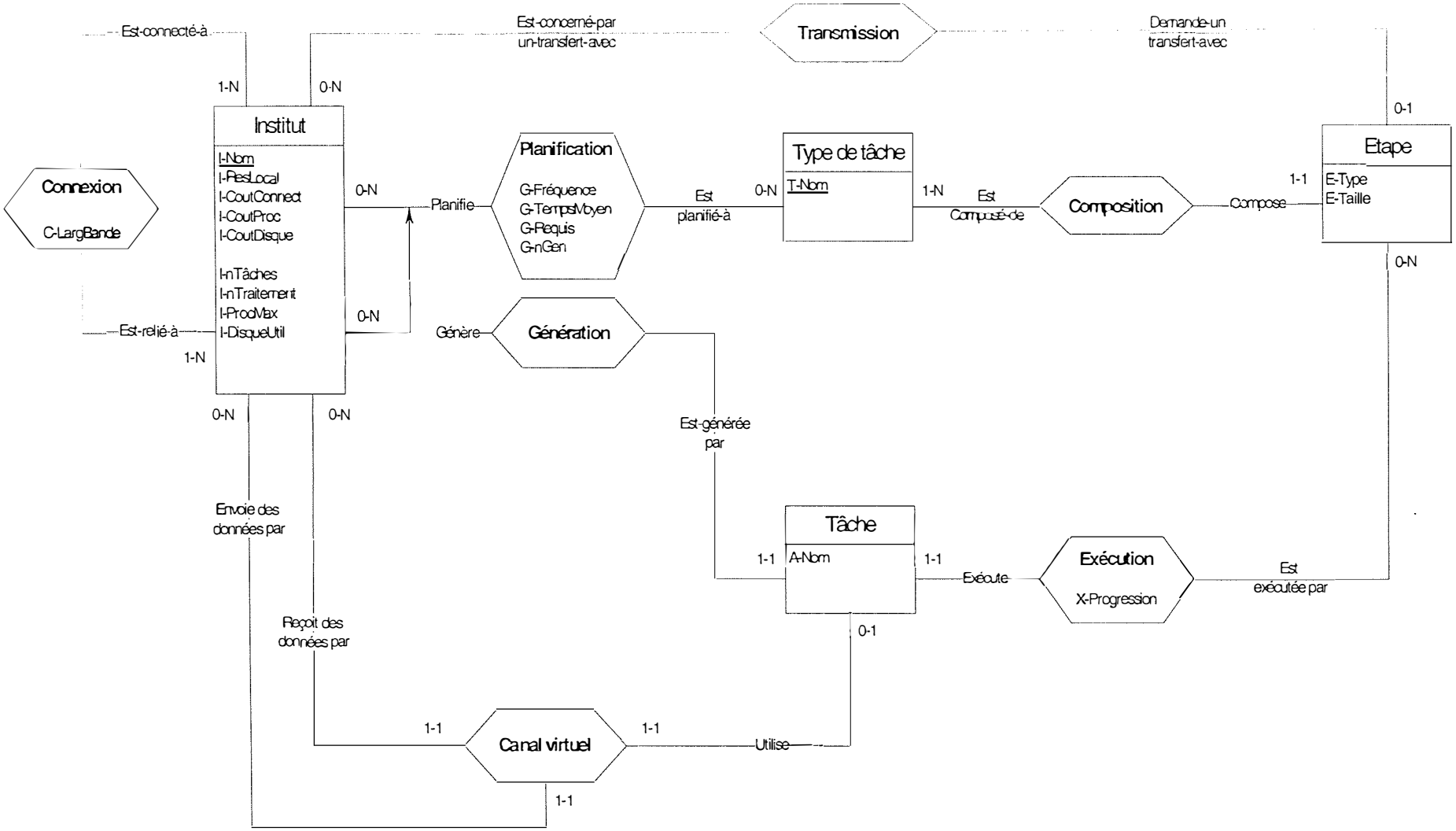


Schéma 4 : Equivalent des tableaux en ERA

Tableau 2 : Les attributs de l'association « Planification »

Nom de l'attribut	Commentaires
G-Fréquence	Fréquence moyenne de génération de la tâche (sec^{-1})
G-Requis	Nombre requis de tâches à exécuter durant la période de simulation (si `G-Fréquence` n'est pas utilisé)
G-TempsMoyen	Temps moyen observé pour exécuter une tâche (sec)
G-nGen	Nombre de tâches générées depuis le début de la simulation

Tableau 3 : Les attributs de l'entité « Etape »

Nom de l'attribut	Commentaires
E-Type	Type de l'étape. Les différents types possibles sont exposés dans la section 2.1.2.
E-Taille	Quantité de données à transférer si l'étape est de type « transfert » (en octets) ou durée du traitement de données si l'étape est de type « traitement » (en sec).

☞ *Validation du schéma ERA*

Pour valider le modèle proposé page 20, il convient d'y ajouter quelques contraintes ad hoc, à savoir :

- ✓ Si un **institut** A *est connecté* à un **institut** B, l'**institut** B doit aussi *être connecté* à l'**institut** A.
- ✓ Un **institut** ne peut *être connecté* à lui-même.
- ✓ Si un **institut** envoie des données par un **canal virtuel**, il ne *reçoit* pas *de données* par ce **canal virtuel.**
- ✓ Si, et seulement si, un **canal virtuel** C permet à un **institut** A d'*envoyer des données* et à un **institut** B de *recevoir des données*, alors :
  - × L'**institut** A *est connecté* à l'**institut** B.
  - × L'**institut** A ou[3] l'institut B *génère* une **tâche** T qui *exécute* une **étape** dont le type `E-Type` est « transfert non local de données » ; cette **étape** *demande un transfert avec* l'autre **institut**. Cette **tâche** T utilise le **canal virtuel** C.

[3] Il s'agit d'un « ou » exclusif.

- ✓ Si, et seulement si, un **institut** *génère* une **tâche** qui *exécute* une **étape** dont le type `E-Type` n'est pas égal à « transfert non local de données », cette **tâche** n'*utilise* pas de **canal virtuel**.
- ✓ Si un **institut** *génère* une **tâche** de nom `A-Nom` égal à N qui *exécute* une **étape** E, cet **institut** *planifie* un **type de tâche** dont le nom `T-Nom` est égal à N et qui *est composée*, notamment, *par* l'**étape** E.
- ✓ Si un **institut** *génère* une **tâche** qui *exécute* une **étape** E, la progression `X-progression` de l'**exécution** de E est inférieure ou égale à la taille `E-Taille` de E.

☞ *Correspondance*

Concrètement, il existe un tableau pour chaque attribut de chaque entité ou association et pour chaque rôle. Par exemple, on aura un tableau de noms d'instituts (un attribut d'entité), un tableau de nombre de types de tâches d'un institut (cardinalité d'un rôle), un tableau de progression des exécutions (attribut d'une association), etc. Une liste des tableaux ainsi qu'une brève description des informations qu'ils contiennent se trouve en annexe 1.

2.2.3 Lecture des données

Les données ne peuvent être entrées que via un fichier « model.dat » ; il n'existe donc aucune interaction directe de l'utilisateur vers le programme. Ce fichier doit être structuré selon une syntaxe précise et très simple que nous ne détaillerons pas ici.

Dans ce fichier, on trouve successivement : le titre du modèle, le nom du fichier de sortie, la capacité des lignes inter-instituts et des réseaux locaux, les informations concernant les instituts (nom, coûts, nombre de types de tâches à générer avec la fréquence respective), les informations concernant les types de tâches eux-mêmes (nom, liste des étapes avec leur type et leur taille). En outre, on peut aussi y trouver quelques informations facultatives comme la durée de la simulation et l'intervalle de temps entre la mesure et l'affichage des différentes variables (nombre de tâches en cours, etc.). Notons qu'une valeur par défaut est utilisée si ces paramètres sont absents du fichier de données.

2.2.4 Simulation proprement dite

☞ ***Méthode de simulation***

La méthode de simulation est relativement simple : une tranche de temps est sélectionnée selon les données du modèle et, après chaque tranche, les différentes variables du simulateur sont mises à jour. De nouvelles tâches peuvent avoir été créées, d'autres se sont éventuellement terminées. Ceci se déroule donc jusqu'à ce que le total des tranches de temps simulées soit égal au temps de simulation demandé. On devine donc bien que ce temps n'a pas la même échelle que le temps réel : en effet, plus la tranche de temps sera petite par rapport au temps de simulation, plus il y aura de calculs et plus la durée d'exécution sera longue... Cela peut aller de moins d'une seconde à plusieurs heures !

☞ ***Choix de la tranche de temps***

Le problème est de choisir entre des tranches très petites, ce qui permet une plus grande précision dans les résultats, ou des tranches plus grandes pour réduire le temps de simulation...

La solution adoptée fut sans conteste la première. En effet, la puissance des processeurs actuels ainsi que la possibilité de faire tourner les simulations en *batch* pendant la nuit donnait moins de poids à l'argument « temps ».

Ainsi, le calcul de la tranche de temps se fit de la manière suivante :

$$\frac{\text{Taille de la plus petite information transférée (octet)}}{\text{Vitesse de transmission de la ligne la plus rapide (octet/sec)} \times 10}$$

Bref, tout transfert de donnée est découpé en au moins dix tranches de temps, ce qui rend les observations et les mesures plus précises et les résultats plus fiables.

Bien sûr, on doit également vérifier que la tranche de temps obtenue est inférieure à la plus petite période des tâches (inverse de la fréquence).

☞ ***Génération de nouvelles tâches***

Comme nous l'avons dit précédemment, à chaque tâche exécutée au sein d'un institut particulier est associée une fréquence de génération.

Cette fréquence peut être exprimée directement (en nombre de tâches par seconde) ou indirectement (en spécifiant un nombre requis de tâches générées durant le

temps de simulation). Par exemple, si ce temps est de 1 heure et si l'on veut qu'une tâche d'un certain type soit générée au CERN toutes les 10 secondes, on peut spécifier une fréquence égale à 0,1 (c'est-à-dire une période de 10 secondes) ou bien exiger qu'au moins 360 tâches soient générées avant la fin de la simulation.

Il est important de préciser que la fréquence de génération des tâches est une variable aléatoire de distribution uniforme dont la moyenne découle directement des chiffres donnés.

De cette fréquence, on peut facilement obtenir une probabilité de génération de tâche par tranche de temps, en faisant le calcul suivant :

```
Pr[1 tâche générée pendant 1 tranche de temps]
= Pr[1 tâche générée pendant 1 s] * tranche de temps (s)
= Fréquence de la tâche (s^-1) * tranche de temps (s)
= P_Tranche
```

Ainsi, après chaque tranche de temps, il suffit de générer un nombre entre 0 et 1 (grâce à un générateur de nombres aléatoires de distribution uniforme présenté au chapitre suivant), puis de le comparer à $P_{Tranche}$: si le nombre généré est inférieur ou égal à $P_{Tranche}$, une nouvelle tâche est créée, dans le cas contraire on attendra la prochaine tranche de temps pour réessayer.

Enfin, dans le cas où un nombre requis de tâches a été spécifié, la fréquence doit être continuellement mise à jour au fur et à mesure que le temps passe. En effet, si 50 tâches doivent être générées pendant 50 secondes et si le hasard a voulu qu'aucune ne soit créée pendant les 5 premières secondes, la fréquence ne peut plus être égale à 1 mais doit être ramenée à 1,11 (= 50 / 45).

- ***Mise à jour des tâches***

Alors que de nouvelles tâches sont éventuellement générées, l'état de celles qui existent déjà est mis à jour, c'est-à-dire que l'on tient compte de la durée de la tranche de temps dans leur progression.

Pour les tâches dont l'étape en cours est de type « traitement de données », on augmente simplement le compteur de temps d'exécution de la valeur de la tranche de temps.

Pour les tâches dont l'étape en cours est de type *transfert de données*, la quantité de données transférées (mesurée en octets) durant la tranche de temps s'obtient en calculant :

$$\text{Quantité de données} = \text{tranche de temps (s)} \times \frac{\text{largeur de bande de la ligne (octet/s)}}{\text{nombre de canaux virtuels sur la ligne}}$$

Si l'étape de mise à jour est terminée, on passe à l'étape suivante s'il y en a une ; s'il n'y en a pas, la tâche est terminée. Le Schéma 5 ci-dessous illustre ce procédé de mise à jour, l'algorithme proposé étant appliqué lors de la mise à jour de chaque tâche en cours d'exécution.

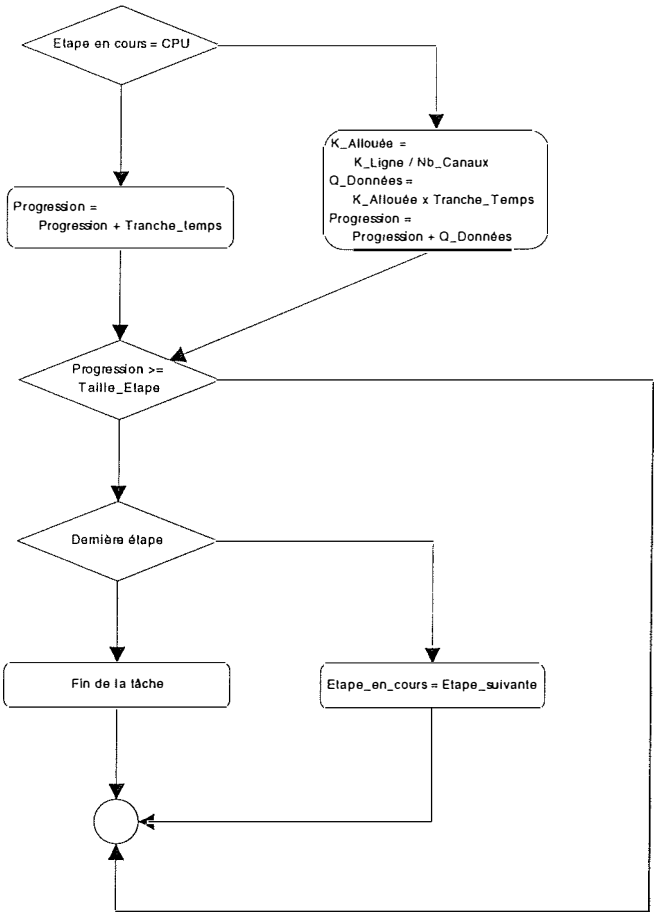


Schéma 5 : Représentation algorithmique de la mise à jour des tâches

2.3 Exemple de session avec le simulateur

Voici un exemple simple mais illustratif d'une session avec ce simulateur. La section 2.3.1 reprend le fichier d'entrée utilisé, la section 2.3.2 présente une partie des résultats obtenus tels qu'ils sont produits par le simulateur et la section 2.4 analyse en détail l'exemple présenté.

Le modèle, de nom « remote analysis », comporte deux instituts (CERN et LYON) reliés par une ligne de 1 Mbps ; au sein du second sont générées 10 occurrences d'une tâche « Analyse event from CERN » comportant trois étapes :

- ✓ Importer 1 MB de données provenant du CERN
- ✓ Traiter des données durant 1 seconde
- ✓ Ecrire 2 MB de données sur le réseau local

Le temps de simulation est fixé à 200 secondes et l'état du système sera affiché toutes les 20 secondes.

2.3.1 Script du modèle (model.dat file)

```
Title:Remote Analysis
#
#debug
# 1
Task:Analyse Event from CERN
Steps:3
Steptype:GET CERN
Stepsize:1.0
Steptype:PROC
Stepsize:1.0
Steptype:PUT LOCAL
Stepsize:2.0
#
Institute:CERN
CostDisk:200.0
CostNet:100000.0
CostCPU:2000.0
#
Institute:LYON
CostDisk:200.0
CostNet:100000.0
CostCPU:2000.0
Tasks in Mix:1
Type:1
Frequency:0.0
Number:10
#
ATM:1.0           (Capacité des lignes du modèle en Mbits/s)
TimeLimit:200.
SnapShot:20.
```

2.3.2 Résultats de la simulation

```
1.      Input Summary for "Remote Analysis"

2.      Total of 1 task types
3.
4.      Analyse Event from CERN  (3 steps):
5.      GET data fro size:    1.000
6.      PROCESS data size:    1.000
7.      PUT data loc size:    2.000

8.      ATM Rate:   1 Mbits/second

9.      Total of 2 institutes

10.     CERN
11.     Disk:     0.2 $ per GByte
12.     WAN:      0.317 $ per line per sec
13.     CPU:      2000 $ per 100 MIPS

14.     LYON (1 task) :
15.     Analyse Event from CERN
16.        freq:        not specified
17.        num:        10

18.     Disk:     0.2 $ per GByte
19.     WAN:      0.317$ per line per sec
20.     CPU:      2000 $ per 100 MIPS

21.     Network Point to Point Rate Matrix ... (MBytes/Sec)

22.     CERN                              0.125    0.125
23.     LYON                              0.125    0.125

24.     LYON: need approx 0 concurrent "Analyse Event from CERN" tasks

25.     Task slice minimum:          20 sec
26.     Smallest data set size is:         1 MBytes
27.     Maximum network bandwidth is:      0.125 MBytes/sec
28.     Simulation over              200 seconds
29.
30.     Time slice will be:          0.8 seconds
```

```
Snapshots

Time: 20.0
   dT:          0.8000000
   VCs:         1
   Tasks:       1
Time: 40.0
   dT:          0.8000000
   VCs:         2
   Tasks:       2

LYON LAN Rate   0.25000 MBytes/sec

Time: 60.0
   dT:          0.8000000
   VCs:         0
   Tasks:       0

Time: 80.0
   dT:          0.8000000
   VCs:         2
   Tasks:       2

LYON LAN Rate   0.12500 MBytes/sec

Time: 100.0
   dT:          0.8000000
   VCs:         3
   Tasks:       3

LYON LAN Rate   0.25000 MBytes/sec

Time: 120.0
   dT:          0.8000000
   VCs:         2
   Tasks:       2

LYON LAN Rate   0.25000 MBytes/sec

Time: 140.0
   dT:          0.8000000
   VCs:         1
   Tasks:       1

LYON LAN Rate   0.12500 MBytes/sec

Time: 160.0
   dT:          0.8000000
   VCs:         2
   Tasks:       2

LYON LAN Rate   0.12500 MBytes/sec

Time: 180.0
   dT:          0.8000000
   VCs:         0
   Tasks:       0

Time : 200.0
   dT:          0.8000000
   VCs:         0
   Tasks:       0

End time:       200 with  0 task(s) executing
```

Final results

Statistics for : *CERN*

Disk occupancy : 10 MBytes
WAN lines used : 1
Processors used: 0 100 MIPS unit
Disk cost : 2.0 $
WAN cost : 0.6 $
CPU cost : 0.0 $

Total cost : 2.6 $

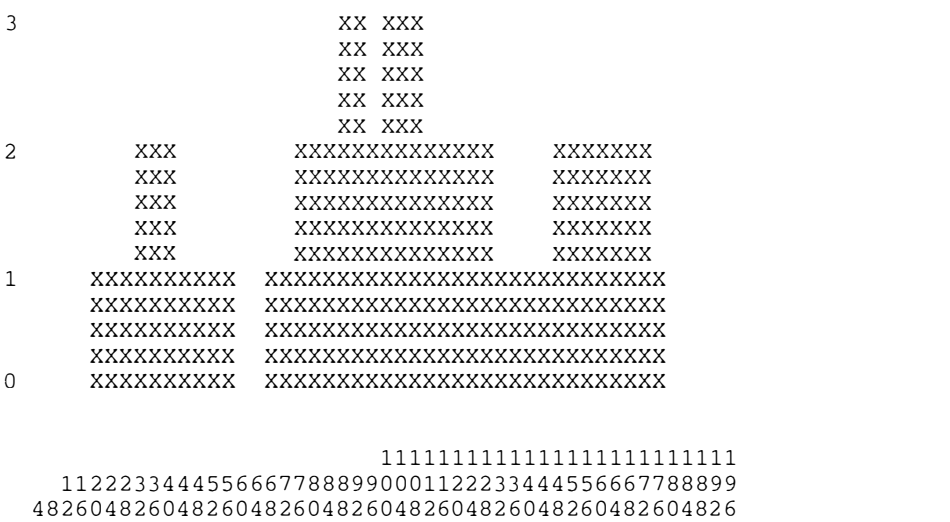
Statistics for : *LYON*

Disk occupancy : 30 MBytes
WAN lines used : 1
Processors used: 1 100 MIPS unit
Disk cost : 6.0 $
WAN cost : 0.6 $
CPU cost : 2000 $

Total cost : 2006.6 $

Task: *Analyse Event from CERN*
Count: 10
Time(ave): 28.16000 sec
I/O Rate(ave): 0.11 MByte/sec

Evolution of tasks at LYON
DATE 15/04/97



Total CPU Cost this Model: 2000.000 $
Total Disk Cost this Model: 8.000000 $
Total WAN Cost this Model: 1.268392 $

Total tasks this Model: 10

Grand Total Cost this Model: 2009.268 $

Finished Simulation
No logical name match

2.4 Analyse des résultats

Analysons en bref le fichier *output* donné par le simulateur... On remarque tout d'abord qu'il se divise en trois parties : un rappel des données, les « snapshots » et les résultats finaux.

2.4.1 Rappel des données

Les lignes 1 à 29 reprennent donc la description du modèle à simuler, à savoir les tâches, les instituts et les capacités de ligne. Un bref descriptif du calcul de la tranche de temps est également spécifié.

2.4.2 Liste des mesures

La rubrique « snapshots » reprend la liste des mesures demandées par l'utilisateur durant la simulation. Dans ce cas précis, elles étaient au nombre de 10, à savoir une mesure toutes les vingt secondes.

Chaque mesure reprend dans l'état en cours du simulateur le nombre de canaux virtuels ouverts (VC's), le nombre de tâches en cours d'exécution (Tasks) et la valeur de la tranche de temps dT (qui peut changer en fonction de l'évolution des tâches). De plus, pour chaque réseau local dont une partie de la largeur de bande est utilisée, il donne une estimation du trafic du moment.

2.4.3 Résultats finaux

La première partie des résultats (lignes 80 à 95) décrit les ressources utilisées par les instituts pour exécuter les différentes tâches demandées. Parmi ces ressources, on retrouve la capacité disque requise (10 MB pour le CERN), le nombre de lignes utilisées (ici une seule) ainsi que le nombre de processeurs qu'il aura fallu pour exécuter les éventuelles tâches « traitement de données » (aucun pour le CERN car pas de tâches de traitement). Pour chaque ressource, une estimation des coûts est également donnée avec un total pour chaque institut. Ces coûts sont en rapport direct avec les coûts unitaires donnés par l'utilisateur et le temps de simulation (par exemple, les coûts d'utilisation des lignes -0.6 $ pour le CERN- sont ramenés sur une période égale au temps de simulation, 200 secondes dans notre cas).

A partir de la ligne 96, on trouve une description de l'exécution de chaque tâche (une dans notre exemple) à savoir le nombre de tâches exécutées (ici 10), le temps moyen d'exécution (ici 28.16 sec) ainsi que le taux de transmission moyen observé pendant les éventuelles étapes de transfert de données (0.11 MBps).

Vient alors une série de « graphiques » qui tentent de donner l'évolution au cours du temps de certaines variables du modèle.

Le plus intéressant d'entre eux (lignes 100 à 123) donne pour l'unique institut exécutant des tâches le nombre de tâches en cours d'exécution. On constate qu'au maximum trois tâches s'exécutaient en même temps et que la simulation s'est terminée alors que toutes les tâches requises étaient terminées. On en déduit que le temps de simulation était insuffisant pour que toutes les tâches s'exécutent l'une à la suite de l'autre.

La dernière partie du fichier (lignes 124 à 130) reprend les totaux relatifs aux coûts et au nombre de tâches exécutées.

2.5 Validation et limitations

2.5.1 Validation

En guise de validation du simulateur, les trois configurations proposées à l'introduction ont été modélisées à l'aide de ce premier outil et les résultats furent présentés par Julian BUNN lors d'une réunion de la commission CMS[4]. Ces résultats, qui ne prennent pas en compte les consultations de données, montrent que le réseau *distribué* coûte globalement trois à quatre fois plus cher que le réseau *centralisé* bien que cette dernière solution soit la plus chère pour le CERN.

Cette information importante est néanmoins à relativiser fortement. En effet, cette conclusion se base uniquement sur le volume des coûts alors que l'auteur même du programme admet que leur évaluation est plus qu'approximative. En outre, il reconnaît qu'il existe toujours un sérieux problème dans l'évaluation des coûts de stockage sur disque (les données qui « transitent » d'un institut à un autre sont supposées stockées de manière permanente) et des coûts de connexion (toute la largeur de bande n'est jamais à 100% utilisable pour les transferts).

[4] Le rapport de cette présentation se trouve en annexe.

Quant à la simulation des transferts de données au travers des différentes lignes du réseau, elle est « théoriquement » correcte. Les résultats sont en effet exacts dans un environnement où les tâches s'exécutent à intervalles relativement constants, sans aucun conflit avec d'autres demandes de transferts effectuées par des tâches extérieures au domaine d'application. Ainsi, tenter de valider les résultats donnés par le simulateur en les comparant au fonctionnement réel d'un réseau conduira inévitablement à un échec. Cet écart consiste néanmoins en un « bémol » supplémentaire à attribuer aux résultats de toute simulation.

2.5.2 Limitations

Cette première version du simulateur suppose donc un environnement assez fermé et très statique. Son plus grand mérite fut de montrer dans quelle mesure les changements de configurations de réseaux ont un impact sur les coûts et les ressources nécessaires. Il a également l'avantage d'être relativement simple et prévisible.

Pourtant, il souffre d'un grand nombre d'inconvénients dont voici les principaux :

- Un transfert de données ne peut être réalisé qu'entre deux instituts adjacents.
- Seules les tâches générées par les instituts occupent les lignes.
- Le concept de « sous-réseau » (nuage) n'existe pas.
- La largeur de bande des lignes doit être la même pour tous les instituts.
- Les processeurs sont supposés être de la même puissance dans tous les instituts.
- Le temps de création et de destruction des canaux virtuels est ignoré.
- La distribution de la fréquence de génération ne peut être que constante.
- Les informations sur la charge des lignes ou le nombre de tâches en cours n'est présentée que de manière agrégée à tous les instituts.
- Le calcul des coûts et l'évaluation des capacités de disque nécessaires manquent de rigueur.

- Un modèle engendrant trop de trafic par rapport à la charge maximale que peuvent supporter les lignes tourne sans problème et sans avertissement à l'utilisateur.
- L'architecture du logiciel rend difficile l'ajout de nouveaux types de tâches ou de nouveaux types de noeuds dans le réseau.
- L'utilisateur a très peu de possibilités d'interaction avec le simulateur.
- L'utilisateur doit encoder les données du modèle dans un fichier d'entrée en respectant un script bien précis.
- Le simulateur ne tourne que sur une plate-forme de type VMS.

C'est pour répondre à la majorité de ces défauts que le projet « Modnet » fut lancé...

3. Validation du générateur de nombres aléatoires

Le simulateur original, tout comme la nouvelle version proposée dans le prochain chapitre, utilise un générateur de nombres aléatoires de distribution uniforme qui a été développé au CERN. La validité des résultats d'un tel simulateur est directement liée à la validité du générateur de nombres aléatoires qu'il utilise. Il est donc primordial dans ce genre de logiciel de s'assurer que le générateur est de bonne qualité.

3.1 Remarque préliminaire

Il existe quelques critères permettant de vérifier si un générateur de nombres aléatoires de distribution uniforme est de bonne qualité ; nous nous intéresserons plus particulièrement aux tests d'uniformité, de corrélation sérielle et à un test visuel.

Ces différents tests seront appliqués à une série de nombres produits par le générateur utilisé dans le simulateur. Ce chapitre a donc un objectif double : illustrer d'une part ces trois méthodes de validation et vérifier d'autre part que le générateur est de qualité suffisante pour ne pas fausser les résultats.

Ce petit exemple de validation s'effectuera sur 20 séries de 1000 nombres produits par le générateur de nombres aléatoires du CERN. Dans chaque série, les nombres ont été ramenés dans un intervalle [0-50[pour faciliter leur ventilation dans 50 classes, à savoir [0,1[, [1-2[, [2-3[, ...

3.2 Les épreuves d'hypothèses

Dans les différents types de tests, on utilisera le principe de « l'épreuve d'hypothèse ». Il permet de tester la validité d'une affirmation relative à une population en se basant sur des observations faites sur un échantillon.

3.2.1 L'hypothèse

L'hypothèse à tester (notée H_0) est donc une propriété généralisée à toute une population à partir d'observations faites sur un échantillon de cette population. Par exemple, on peut poser : « La taille moyenne des hommes dans le monde est de 1 m 72. » ou encore « Il n'existe aucune corrélation entre la taille des Belges et leur consommation de boeuf. » en ne se basant que sur les valeurs observées suite à une enquête sur 1000 personnes.

3.2.2 La région critique

Dans l'espace des échantillons, on déterminera une *région critique* que l'on peut définir comme étant l'ensemble des échantillons défavorables à H_0, c'est-à-dire qui contredisent l'hypothèse.

Par exemple, lors de l'évaluation de l'hypothèse « La taille moyenne d'un être humain est de 1m72 », si l'échantillon sélectionné pour tester la validité de cette hypothèse consiste en un millier de Pygmées, il fera certainement partie de la région critique associée à l'hypothèse, car la moyenne est sensiblement inférieure à 1 m 72.

Le tout est donc de définir précisément quelles sont les limites de la région critique et dans quelle mesure les résultats obtenus à partir de l'échantillon sont suffisamment distants de l'hypothèse pour que celle-ci soit effectivement rejetée. En effet, ce n'est pas parce que l'échantillon donne une moyenne de 1m74 que l'hypothèse de départ doit être considérée comme fausse.

3.2.3 L'erreur de première espèce

Il convient également de donner une certaine probabilité α que l'hypothèse soit rejetée à cause d'un mauvais échantillon alors que celle-ci était vraie. Cette probabilité est également appelée *erreur de première espèce.*

3.2.4 Principe du test

Le test se base sur le concept de la région critique. Si l'échantillon est répertorié comme faisant partie de la région critique, l'hypothèse est considérée comme fausse. Par contre, s'il n'en fait pas partie, il n'est pas possible d'affirmer qu'elle était vraie... mais elle n'est pas rejetée. Le Schéma 6 ci-dessous illustre ce principe dans le cadre du test d'uniformité.

3.3 Test d'uniformité

3.3.1 Principes

Le test d'uniformité permet de vérifier si la répartition des nombres générés se fait uniformément dans les différentes classes possibles. Dans notre cas, vu que 1000 nombres sont générés et qu'ils sont répartis en 50 classes, l'idéal serait d'avoir exactement 20 nombres dans chacune d'entre elles... ce qui est évidemment peu probable. Bref, il faut définir formellement dans quelle mesure cette répartition peut être considérée comme uniforme ou non.

3.3.2 La région critique

La région critique correspond ici à l'ensemble des échantillons dont la répartition est fort « distante » de la répartition parfaite ou *théorique* (20 nombres par classe). Il faut donc déterminer, par rapport à la répartition théorique, la valeur de cette *distance* (notée A) qui délimite la région critique. Ainsi, en calculant cette même distance pour la répartition observée, on peut facilement déterminer si l'échantillon choisi fait partie ou non de la région critique comme l'illustre le Schéma 6 ci-dessous.

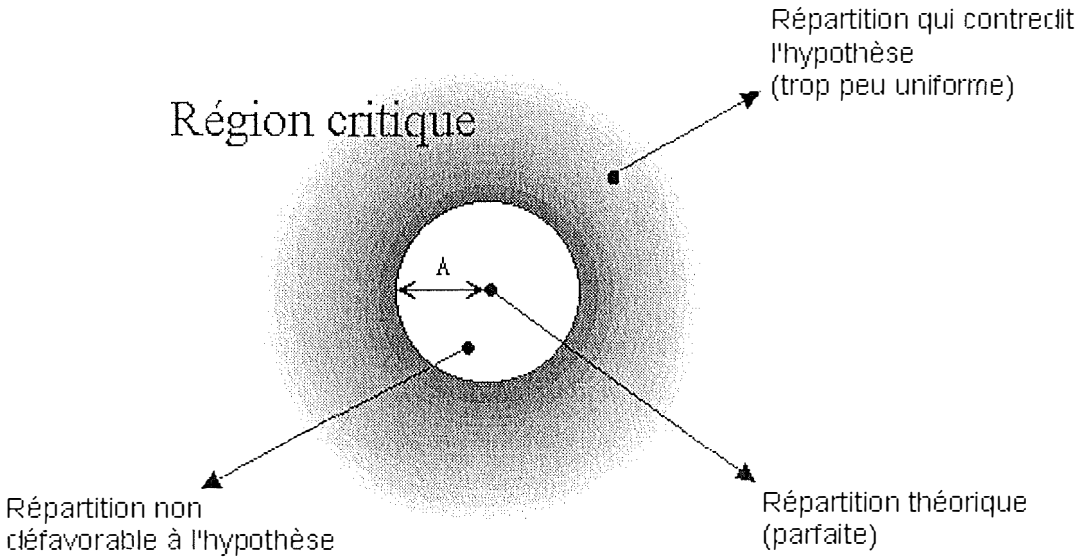


Schéma 6 : La région critique

3.3.3 Calcul de la distance observée.

Le comptage des nombres dans les différentes classes nous donne un vecteur de répartition observée (R_{obs}). Il s'agit donc de calculer la distance (notée D^2) entre ce vecteur et le vecteur de répartition théorique (R_{Th}) dont toutes les composantes valent 20.

Ainsi,

$$D^2 = \sum_i \frac{(R^i_{Obs} - R^i_{Th})^2}{R^i_{Th}}$$

3.3.4 Calcul de la distance délimitant la région critique.

La distance calculée ci-dessus devra être comparée à une certaine valeur A au-delà de laquelle elle sera considérée comme trop grande (et l'hypothèse rejetée).

L'expression du calcul de la valeur de A peut être facilement mise en évidence en partant de α qui, rappelons-le, est la probabilité de rejeter H_0 alors qu'elle était vraie.

On a donc :

$$\begin{aligned} \alpha &= Pr\,[D^2 > A \mid H_0] \\ &= Pr\,[D^2 > A \text{ et } H_0] \,/\, Pr[H_0] \qquad (H_0 \text{ sup. vraie}) \\ &= Pr\,[D^2 > A] \\ &= 1 - Pr\,[D^2 \leq A] \end{aligned}$$

Affirmons, sans le démontrer, que la distribution de D^2 tend vers une distribution Khi-carré dont le degré de liberté est égal au nombre de classes moins un lorsque la taille de l'échantillon tend vers l'infini. Ce résultat est dû à K. Pearson qui a donné son nom à ce type de test. En pratique, pour une taille finie, on admet la validité de l'approximation Khi-Carré lorsqu'il y a au moins 5 observations *théoriques* dans chaque classe ($R_{Th} > 5$).

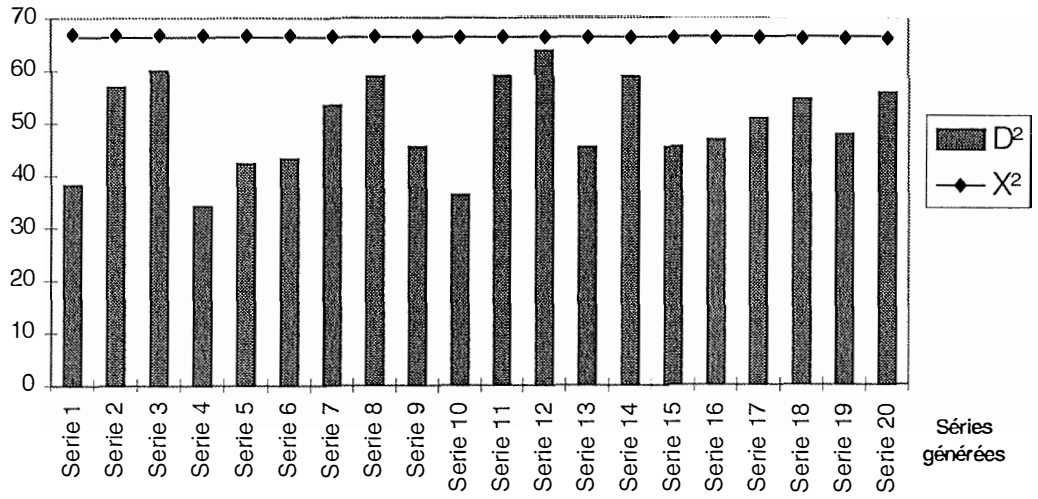
Donc,

$$\begin{aligned} \alpha &= 1 - F_{Khi^2}(A) \\ \Leftrightarrow F_{Khi^2}(A) &= 1 - \alpha \\ \Leftrightarrow \quad A &= Q_{khi^2}(1 - \alpha) \qquad (D° \text{ de liberté} = 19) \end{aligned}$$

En conclusion, la répartition des nombres générés ne sera pas considérée comme uniforme si $D^2 > Q_{khi^2}(1 - \alpha)$.

3.3.5 Calculs et résultats

Le graphique 1 ci-dessous compare, pour chaque série de 1000 nombres générée, la valeur observée de D^2 et la valeur théorique (et donc constante car indépendante de l'échantillon) X^2. Notons que la probabilité α a été fixée à 5%.



Graphique 1 : Comparaison entre D^2 et X^2

3.3.6 Conclusion

On constate rapidement que les résultats sont très satisfaisants : aucun échantillon n'est rejeté alors qu'on aurait pu s'attendre à 5% d'erreur (cfr. α) c'est-à-dire 1 mauvaise série parmi les 20 générées. On peut considérer sans trop se tromper que le générateur assure une bonne uniformité dans la production de nombres aléatoires. L'hypothèse de départ n'est donc pas rejetée.

3.4 Test de corrélation sérielle

3.4.1 Principes

Ce test permet d'étudier d'une part si les différents nombres générés dans une même série sont corrélés entre eux (et cela à différents intervalles) et d'autre part s'il existe une corrélation entre les nombres générés par deux séries différentes. Par exemple, la série présentée ci-dessous est corrélée notamment à l'intervalle 8 où chaque nombre est égal à son « précédent » plus un.

1 8 4 6 3 4 9 1 2 9 5 7 4 5 10 2 3 10 6 8 5 6 11 3 4 11 7 9 6 7 12 4

3.4.2 La région critique

La région critique correspond à l'ensemble des échantillons dont la corrélation est fort « distante » de la corrélation nulle.

Comme dans la méthode précédente, l'important est de déterminer la valeur de la *distance* qui délimite la région critique pour pouvoir établir ou non si l'échantillon en fait partie.

3.4.3 Calcul de la corrélation observée

De façon générale, si $\overline{x}$ et $\overline{y}$ sont respectivement les moyennes des n valeurs des variables aléatoires x et y, la formule qui permet de calculer la corrélation s'écrit de la manière suivante :

$$corrélation\ (x,\ y) = \frac{covariance\ (x,\ y)}{\sigma_x . \sigma_y}$$

avec :

$$covariance\ (x,\ y) = \frac{\sum_i (x_i - \overline{x}).(y_i - \overline{y})}{n}$$

et :

$$\sigma_x^2 = \frac{\sum_i (x_i - \overline{x})^2}{n} \quad , \quad \sigma_y^2 = \frac{\sum_i (y_i - \overline{y})^2}{n}$$

La corrélation théorique étant égale à la corrélation nulle, on peut considérer la corrélation observée comme la *distance observée.*

3.4.4 Calcul de la distance délimitant la région critique.

Comme dans les deux tests précédents, la distance calculée ci-dessus devra être comparée à une certaine valeur A au delà de laquelle elle sera considérée comme trop grande (et l'hypothèse rejetée).

Connaissant la probabilité α qu'un échantillon valable soit rejeté, on peut calculer A :

$$\alpha = Pr[\rho_{xy} > A \mid H_0]$$

$$= Pr[\rho_{xy} > A \text{ et } H_0] / Pr[H_0] \qquad (H_0 \text{ sup. vraie})$$

$$= Pr[\rho_{xy} > A]$$

$$= 1 - Pr[\rho_{xy} \leq A]$$

La distribution de ρ_{xy} est une distribution difficile à mettre en évidence. Ainsi, on peut la remplacer, sans changer la forme de la distribution, par une autre expression plus facilement identifiable (distribution t-Student de degré de liberté *d*) :

$$\alpha = 1 - Pr\left[\sqrt{\frac{\rho_{xy}^2 . d}{1-\rho_{xy}^2}} \leq A\right]$$

$$= 1 - F_{t\text{-}Student(d)}(A)$$

Donc,

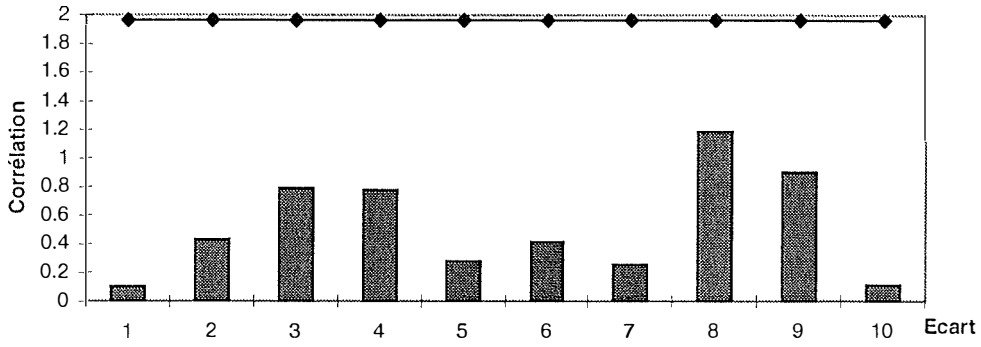
$$F_{t\text{-}Student(d)}(A) = 1 - \alpha$$

$$\Leftrightarrow \quad A = Q_{t\text{-}Student(d)}(1 - \alpha)$$

En conclusion, la corrélation sera jugée trop importante si sa valeur modifiée (expression en racine carrée) dépasse $Q_{t\text{-}Student(d)}(1 - \alpha)$.

3.4.5 Calculs et résultats

Le graphique 2 ci-dessous compare, pour une série de 1000 nombres, la corrélation observée et la valeur théorique. Notons que la probabilité α a été fixée à 5% et que le degré de liberté de la distribution de Student est égal à la taille de l'échantillon moins la taille de l'intervalle. Le tableau 3 reprend les valeurs utilisées dans le graphique 2 ainsi que la corrélation réelle.



Graphique 2 : Comparaison des corrélations avec la valeur théorique *t-Student*

Ecart	$\rho_{x\,x+i}$	$\sqrt{\ldots}$	t-Student
1	0.00323502	0.10224974	1.96234396
2	0.01360739	0.4299127	1.96234396
3	0.02502513	0.79042368	1.96234396
4	0.02462669	0.77744095	1.96234851
5	0.00874329	0.27580542	1.96235305
6	0.01307844	0.41236929	1.96235305
7	0.00804821	0.25362255	1.9623576
8	0.0376135	1.1855148	1.9623576
9	0.02863725	0.90187473	1.96236215
10	0.00357495	0.11248377	1.96236215

Tableau 4 : Résultats du test de corrélation au sein d'une série de 100 nombres

3.4.6 Conclusion

Comme dans le test précédent, la distance observée est suffisamment petite pour conclure au non-rejet de l'hypothèse de départ. Bien sûr, le test n'étant réalisé que sur des intervalles allant de 1 à 10, rien ne permet d'affirmer qu'une certaine corrélation n'existe pas à un intervalle plus grand. Néanmoins, les résultats ci-dessus laissent à penser que les séries de nombres produites par le générateur ne sont pas corrélées.

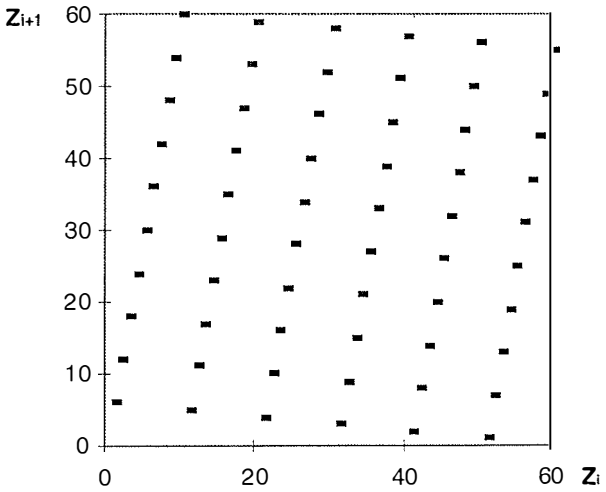
3.4.7 Tests visuels de corrélation

La corrélation entre deux séries de nombres peut également être mise en évidence en représentant les paires (x_i , y_i) par des points sur un écran (par exemple un diagramme cartésien à deux dimensions), ces points doivent présenter une distribution uniforme sur toute la surface de l'écran. Si des raies apparaissent, cela signifie que certaines corrélations existent.

Par exemple, le générateur de Lehmer [NOIR97] est un générateur de nombres entiers défini comme suit (a, m et z_0 sont des paramètres du générateur ; m est premier):

$$Z_{n+1} = (a.Z_n) \bmod m$$

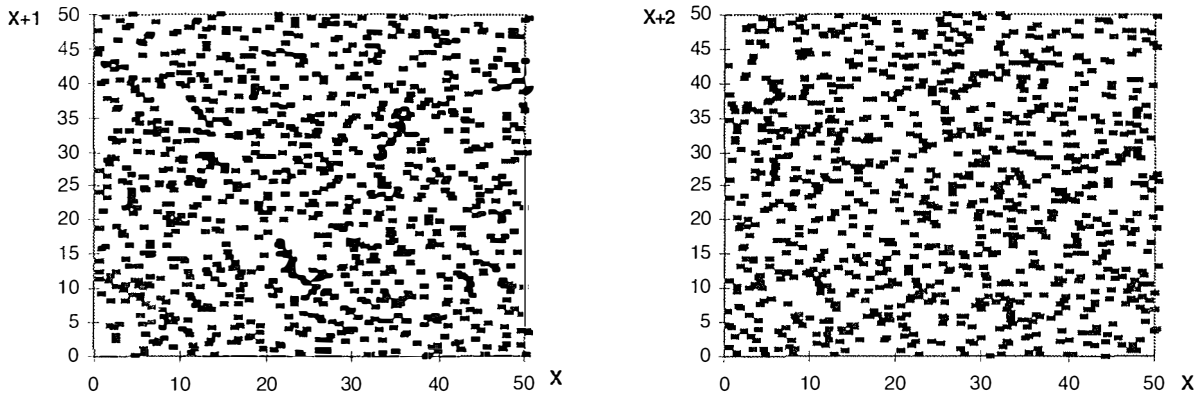
On peut facilement constater que la période maximum observable dans une liste de nombres générés est égale à m. Ainsi, dans le graphique 3, on constate qu'il existe une corrélation évidente entre Z_i et Z_{i+1} malgré le caractère *cassant* de la fonction modulo.[5] Pour information, 1000 nombres ont été générés et les paramètres du générateur étaient : a = 6 , m = 61, z_0 = 1.



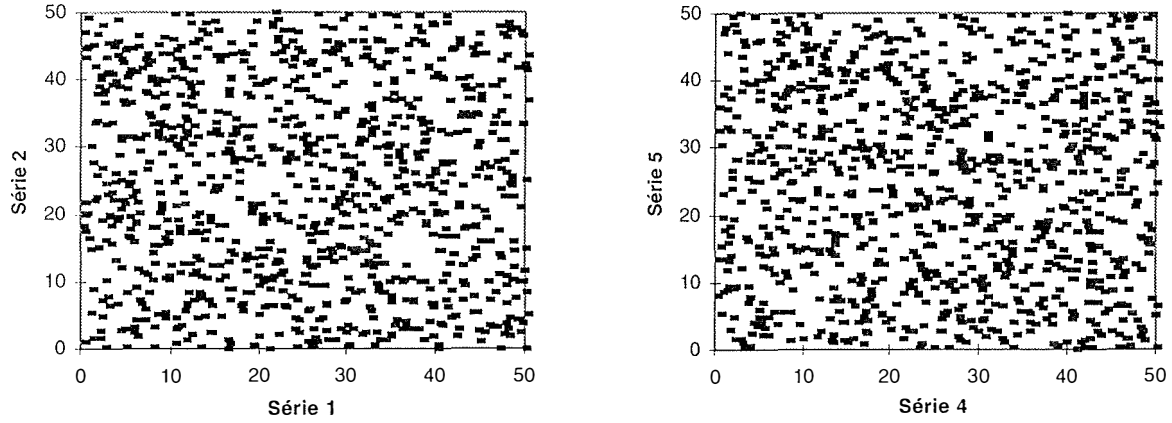
Graphique 3 : Corrélation dans une série de Lehmer

[5] Il va de soi que dans ce genre de générateur, il existe par définition toujours une corrélation entre un nombre et son précédent. Néanmoins, on parle ici d'une corrélation sensible observable entre les différents nombres générés.

Quant au générateur du CERN, le Graphique 4 montre deux exemples de test visuel entre les nombres d'une série avec successivement un intervalle de une (x_i, x_{i+1}) et deux (x_i, x_{i+2}) unités. Le Graphique 5, quant à lui, effectue le même test avec les nombres de deux séries différentes (x_i, y_i). Aucun ne décèle de « raie » ce qui permet de conclure à l'absence de corrélations flagrantes.



Graphique 4 : Tests visuels de corrélation intra-série



Graphique 5 : Tests visuels de corrélation entre séries

3.5 Conclusion

Le générateur de nombres aléatoires de distribution uniforme utilisé dans les différents simulateurs passe avec succès les tests d'uniformité et de corrélation sérielle ainsi que le test visuel. On peut donc dorénavant supposer que les résultats fournis par les simulateurs qui se basent sur ce générateur ne seront pas entachés d'erreurs dues à un mauvais générateur de nombres aléatoires.

4. Projet « Modnet » : un nouveau simulateur

Après présentation du simulateur auprès d'une commission d'évaluation au CERN, il fut officiellement établi qu'il existait bien là un potentiel intéressant à explorer. Dès lors, un développement approfondi du modèle et de l'interface fut demandé. Ainsi allait naître le projet « Modnet », la deuxième génération de simulateur de réseaux.

4.1 Contexte et philosophie générale

4.1.1 Objectifs

Le projet « Modnet » est donc né en réponse aux limitations décrites à la section 2.5. Les objectifs de cette deuxième version du simulateur peuvent se résumer en trois grandes catégories :

- Raffinement du modèle de simulation
- Création d'une architecture orientée objet flexible
- Amélioration de l'interface homme-machine.

☞ ***Raffinement du modèle de simulation***

Le modèle utilisé dans le premier simulateur n'étant pas adéquat pour représenter les différentes configurations possibles de réseaux, et les informations fournies pendant et après la simulation étant relativement restreintes, il s'avère utile d'apporter un nouveau souffle au logiciel. La section 4.2 présente les différentes idées qui ont été conservées dans la nouvelle version ainsi que les nouveautés qui enrichissent son modèle.

☞ ***Création d'une architecture orientée objet flexible***

En réponse au premier simulateur dont la structure ne consistait qu'en une seule fonction riche en branchements et en code recopié, Modnet revêtira une structure entièrement orientée objet. La section 4.3 discutera d'une part de la justification de ce choix ainsi que des avantages qu'il apporte et, d'autre part, de l'architecture OO de Modnet.

☞ ***Amélioration de l'interface homme-machine***

La première version du simulateur était essentiellement un prototype utilisé presque exclusivement par son développeur, Julian BUNN. Modnet étant destiné à être utilisé par d'autres personnes, une interface homme-machine plus riche se justifiait. La section 4.4 montrera les principes fondamentaux d'une bonne interface et les illustrera en présentant les différents objets graphiques de Modnet.

4.1.2 Moyens techniques

Modnet a été développé dans l'environnement Windows 95 en utilisant les librairies MFC[6] de Microsoft Visual C++ 4.2. Ces librairies offrent de nombreux objets graphiques ainsi que des outils de développement très puissants comme le « Class Wizard » qui permet la gestion et la maintenance de tous les objets du logiciel.

Le choix de cet environnement permet de bénéficier de nombreux avantages. Outre le fait que cette plate-forme soit fort répandue au CERN, elle permet le développement rapide d'applications OO 32 bits munies d'une interface riche. La possibilité de partager les objets du logiciel avec d'autres applications en utilisant la technologie OLE et de développer aisément une version « internet » du programme sont d'autres atouts qu'offre l'environnement Visual C++ sous Windows 95.

4.1.3 Taille du projet

Le projet « Modnet » s'est étalé sur une période de six mois, d'août 1996 à janvier 1997 et a été développé au CERN qui reste propriétaire du logiciel. Deux personnes travaillaient directement sur le logiciel : Julian BUNN, l'auteur du premier simulateur, dont le rôle consistait à superviser, orienter et valider le développement du programme, et moi-même qui, en tant que stagiaire à temps plein, ai implémenté la totalité du simulateur, après avoir procédé à une analyse de la version existante (en Fortran) et établi un patron de l'architecture OO.

4.2 Idées reprises et nouveautés

Modnet ne fait pas table rase des différents concepts introduits dans la première version du simulateur. Il reprend en effet les différents types d'informations manipulés ainsi que l'idée de « tranche de temps ». A cela sont ajoutées un grand nombre de nouveautés qui complètent, enrichissent et parfois corrigent le modèle de départ.

4.2.1 Idées reprises

Toutes les informations manipulées dans la version initiale du simulateur se retrouvent dans Modnet. Ainsi, les instituts, les tâches découpées en étapes et les coûts unitaires y trouvent leur équivalent. Les différents types d'étapes sont également repris

[6] Microsoft Foundation Classes

excepté les transferts de données avec un institut choisi au hasard qui ont été jugés inutiles dans le cadre des simulations envisagées au CERN.

L'idée de « tranche de temps » fait aussi partie de Modnet. Calculée de la même façon, la tranche de temps correspond toujours à l'intervalle de temps entre chaque mise à jour des différentes variables du modèle.

4.2.2 Les nouveautés

Outre la nouvelle interface et l'environnement Windows 95, le logiciel dispose de beaucoup de nouvelles idées. Certaines ont un rapport direct avec la simulation, d'autres consistent en de nouveaux outils ou services intéressants.

Dans la première catégorie, on trouve donc une série de nouveautés qui enrichissent le modèle :

- Un nouveau type de noeud fait son apparition. Il s'agit du noeud « Nuage » qui consiste en un sous-réseau dont les noeuds qui le composent n'ont pas de véritable intérêt dans le contexte de la simulation. Les informations échangées entre les différents instituts peuvent transiter par ces nuages en un certain temps calculé à partir de leurs propriétés (cfr. section 4.3.2).
- Des informations peuvent à présent être échangées entre deux instituts qui ne sont pas physiquement directement connectés.
- Le concept de ligne est explicitement utilisé. En effet, dans la précédente version du simulateur, le fait qu'un institut envoie des données à un autre suppose qu'une ligne les reliait physiquement ; de même, une ligne n'existait entre deux instituts que s'ils s'échangeaient des données. Ainsi, la structure réelle du réseau n'existait pas puisqu'elle changeait selon les tâches. A présent, la structure du réseau est construite en dehors de toute simulation (ensemble de noeuds et de lignes physiques) et c'est sur cette structure que circuleront les données, certaines lignes pouvant ne pas être utilisées.
- Les informations échangées entre les instituts suite aux tâches qu'ils ont générées ne sont pas les seules données qui circulent sur les lignes. Ainsi, à chaque ligne est associée un certain trafic de base (*background traffic*) qui peut être constant ou varier aléatoirement en suivant une loi de distribution statistique.

- La largeur de bande disponible pour les données peut être réduite par l'utilisateur. En effet, selon le protocole de communication utilisé, une certaine proportion de la largeur de bande est utilisée par la signalisation.
- Chaque ligne peut être à tout moment désactivée (puis réactivée) par l'utilisateur pour simuler un problème au niveau physique.
- Une nouvelle rubrique est ajoutée dans le calcul des coûts : les coûts de signalisation. Ceux-ci sont proportionnels au nombre de connexions demandées depuis chaque institut.
- La fréquence de génération des tâches peut également suivre une distribution autre qu'uniforme. Par exemple, on peut spécifier une fréquence qui varie selon une distribution normale dont la moyenne et l'écart-type sont fixés par l'utilisateur.
- Chaque institut peut avoir des processeurs d'une puissance qui lui est propre. De plus, l'utilisateur peut fixer le nombre maximum de processeurs qu'un institut peut allouer aux tâches de la simulation.
- Chaque institut dispose d'une certaine quantité d'informations avant la simulation. Ces informations sont stockées sur disque et sont donc comptabilisées dans la capacité nécessaire lors de l'évaluation des coûts.

La seconde catégorie reprend donc les idées qui ne modifient pas de manière sensible la simulation mais qui apportent à l'utilisateur de nouvelles possibilités et de nouveaux outils. On trouve les nouveautés suivantes :

- Les informations données pendant et après la simulation sont plus riches. Ainsi, on pourra connaître à tout moment l'état d'avancement de la génération des tâches (tâches générées, tâches terminées, fréquence), le nombre de processeurs utilisés et le temps moyen d'exécution. Pour chaque ligne, on peut également mesurer le trafic (décomposable en « background traffic » et en trafic engendré par les tâches générées) et le nombre de connexions ouvertes.
- Chaque donnée peut être affichée sous forme de graphique ou de liste de valeurs, la fréquence des mesures étant spécifiée par l'utilisateur. Afin de pouvoir comparer facilement plusieurs informations, plusieurs graphiques peuvent être affichés en même temps, dans une même fenêtre ou dans des fenêtres séparées. Le choix des échelles est automatique bien que la possibilité soit laissée à l'utilisateur de fixer sa propre échelle.

- L'unité choisie pour spécifier la taille des étapes peut être choisie parmi des unités de base[7] ou bien être construite à partir d'autres unités. Par exemple, si l'on veut envoyer 2 Mo de données qui correspondent à quatre photos de taille identique représentant une collision de particules, on peut définir une unité « Image de collision » de 500 Ko et spécifier une étape dont la taille est de « 4 images de collision ».
- A tout moment, l'utilisateur peut demander un rapport des valeurs de chaque variable du modèle. Ce rapport peut être affiché et/ou être sauvé dans un fichier.
- Le réseau construit peut être imprimé tel qu'il apparaît à l'écran.

4.3 L'architecture OO de Modnet

4.3.1 Pourquoi une architecture OO ?

☞ ***Comment le paradigme OO est-il né ?***

Le développement d'applications importantes en C, pascal, fortran, cobol ou tout autre langage de programmation classique a mis en avant des difficultés telles que la maintenance, la réutilisation et la fiabilité des programmes. Il s'agit des préoccupations majeures du génie logiciel, qui a développé, dans le but de les maîtriser, des langages de plus haut niveau.

La base de ces langages a été la possibilité de considérer des éléments de programme comme des entités autonomes ne communiquant qu'à travers une interface externe réduite. L'intérieur des composants est alors vu par les autres comme une boîte noire. C'est l'introduction des concepts de modularité et d'encapsulation.

Plus récemment, l'augmentation du nombre de développements de logiciels nouveaux a centré l'attention sur un autre point : la réutilisation des logiciels. En effet, il est apparu que, lors de la réalisation d'un logiciel, une part importante de travail consistait à réécrire des fonctions et des modules déjà existants mais non complètement adaptés à l'application visée. Compte tenu du coût de la programmation, l'écriture répétée d'éléments fonctionnels semblables est apparue

[7] *Octet*, *Ko*, *Mo* ou *Go* pour les étapes de transferts de données ; *seconde*, *minute* ou *heure* pour les étapes de traitement de données.

comme un handicap majeur. Les concepts d'encapsulation et de modularité ont alors été repris et développés.

Par ailleurs, l'analyse du cycle de vie des logiciels a fait apparaître qu'en général les éléments les plus stables, au moins d'un point de vue sémantique, étaient les structures de données, dont les variations portaient essentiellement sur la modification ou l'ajout des traitements.

Cette analyse a alors introduit l'idée de centrer la structuration des programmes autour des données en leur associant intimement les traitements qui leur sont spécifiques plutôt que, comme précédemment, autour des activités, fonctions qui sont amenées à être modifiées plus fréquemment. C'est le principe de la programmation orientée objet. L'intégration des concepts de modularité, encapsulation et généricité a permis la définition de langages OO à part entière : centrage autour des données, accès limité (encapsulation) et généricité (héritage).

☞ *Objectifs de la programmation OO*

On peut considérer que l'objectif de la programmation orientée objet est essentiellement d'améliorer la qualité des logiciels en programmant moins et mieux.

Pour moins programmer, il faut disposer de langages de plus haut niveau que les langages classiques et qui se rapprochent autant que possible des étapes d'analyse et de spécification des programmes. Il faut également que ces langages permettent d'éviter le redéveloppement complet de programmes ou de modules déjà existants lorsque seules quelques modifications sont apportées.

Pour programmer mieux, il faut que ces langages offrent des structures permettant de garantir la validité, la fiabilité et la performance des logiciels développés.

Différents facteurs influent sur la qualité de la programmation et permettent d'en diminuer le volume. En particulier [FERR94]:

- **La portabilité** : C'est la possibilité d'utiliser le même texte de programme pour réaliser une même application, mais sur des machines ou des environnements différents. C'est un des facteurs les plus classiques de la programmation. L'emploi des langages OO permet d'isoler les détails de réalisation dans les niveaux les plus bas de spécification des objets. Ainsi, lors du portage, tous les niveaux supérieurs restent inchangés et l'attention

peut se concentrer sur les points spécifiques du portage. De plus, l'emploi du C++ qui produit du code C permet d'avoir accès aux propriétés de portage de ce dernier.

- **La compatibilité** : Les nouveaux programmes développés doivent être intégrés au sein d'applications déjà existantes. Dans cet objectif, de nombreux langages OO ont choisi d'être associés à un précompilateur qui produit du code source dans un langage procédural classique.

- **La validité** : Les programmes doivent assurer exactement les fonctions définies par le cahier des charges. Ce point est trop souvent laissé à la seule responsabilité du programmeur qui doit s'efforcer de « bien programmer ». En fait, lors de la spécification des programmes, des propriétés de base peuvent être mises en avant pour certains éléments du programme, par exemple : une valeur absolue doit toujours être positive, etc. Dans ce cas, l'emploi du programme avec des valeurs non prévues induit un comportement non valide. Certains langages OO (par exemple le langage Eifel) autorisent la spécification de préconditions, postconditions et axiomes (ou invariants) qui sont des propriétés intrinsèques des données et traitement associés.

- **L'intégrité** : C'est la faculté du programme de protéger son code et ses données contre des accès non autorisés. Des travaux dans ce sens commencent à apparaître, en particulier en ce qui concerne les bases de données OO. Toutefois, les langages OO par eux-mêmes n'offrent pas encore de réponse à ce problème.

- **La fiabilité** : Il s'agit ici d'assurer qu'en cas de défaut de fonctionnement d'un composant de l'application, les conséquences sur l'ensemble ne seront pas catastrophiques. Dans la pratique, les propriétés de modularité, d'encapsulation et de limitation d'accès offertes par ces langages OO permettent d'augmenter de manière significative la fiabilité des programmes. En effet, en cas d'accès indésirable, le compilateur refuse de compiler le programme ; toutefois, il n'y a pas de contrôle d'accès dynamique, et des accès interdits, tels que les débordements de tableaux, s'ils ne sont pas prévus lors de la conception, posent toujours autant de problèmes.

- **La maintenabilité** : La correction et la modification des programmes en cours d'utilisation sont des points essentiels pour les réalisations professionnelles. Ici interviennent la lisibilité, la compréhension et la

structuration des programmes. L'approche très modulaire des langages OO est une bonne réponse si, du moins, l'environnement de développement est muni d'outils de visualisation et de suivi adéquats.

- **La réutilisabilité** : C'est le point sur lequel les gains même minimes auront des répercussions économiques importantes. Il s'agit de pouvoir réutiliser des développements réalisés pour une application dans d'autres contextes. Cela nécessite d'être pris en compte dès la conception et d'effectuer un effort supplémentaire lors du développement afin de garantir une certaine généralité aux modules développés. L'emploi de l'héritage pour spécifier et particulariser progressivement des modules en fonction de l'application particulière apparaît comme une solution très prometteuse. Dès aujourd'hui, des bibliothèques d'éléments généraux sont proposées au public, typiquement des modules (ou « classes » pour les langages OO) permettant, par exemple, des traitements graphiques ou plus classiques comme le tri.
- **L'extensibilité** : C'est la faculté de pouvoir adjoindre des données ou fonctionnalités supplémentaires sans corrompre l'intégrité du logiciel préexistant. Les notions de limitation d'accès et d'héritage sont de bonnes réponses offertes par les langages OO à ce problème.
- **L'efficacité** : C'est la possibilité d'exploiter au mieux les ressources offertes par la ou les machines où le logiciel sera implanté. Le langage C est en général considéré comme très efficace, il en est de même pour le C++, mais ces langages, comme les autres, objets ou non, sont encore loin d'offrir toutes les facilités souhaitables d'exploitation des ressources matérielles. Mais, à vrai dire, ce facteur d'efficacité est bien souvent incompatible avec d'autres facteurs comme la portabilité ou l'extensibilité.

Pour conclure l'énoncé de ces différents facteurs, il faut souligner que globalement la programmation OO, de part sa structuration plus fine, ses propriétés d'encapsulation (accès aux informations limité ou interdit), d'héritage (réutilisation) et de généricité (utilisation avec différents types de données), semble apporter une réponse nettement plus satisfaisante que celle de la programmation classique.

☞ *Ce choix appliqué à Modnet*

Indépendamment de l'effet de mode produit par la programmation orientée objet, c'est ce paradigme qui fut préféré pour implémenter Modnet. Parmi les

nombreux avantages listés ci-dessus, la notion de réutilisabilité s'avéra être un critère prépondérant. En effet, si l'on désire ultérieurement ajouter de nouveaux types d'étapes, par exemple des étapes de traitement de données de durée aléatoire, ou de nouveaux types de noeuds, par exemple des instituts qui ne peuvent que recevoir des données, il faut s'assurer que les modifications à apporter au code du programme soient minimes et que tout le code déjà écrit puisse être réutilisé. Pour favoriser cette réutilisabilité, certains sacrifices ont dû être fait au détriment de l'efficacité... mais au profit de la généralité. Le programme compte par exemple de nombreux types de listes d'objets ; il aurait certainement été efficace d'implémenter une liste par type d'objet, taillée sur mesure. Pourtant, dans un souci de clarté et de généralité, une liste abstraite composée d'éléments abstraits fut l'objet de base pour implémenter toutes les listes du logiciel.

D'autres critères ont indirectement agi sur le choix du paradigme de programmation. On peut entre autres citer le fait que les différents éléments graphiques (fenêtres, boutons de commande, menus, ...) manipulés par l'interface homme-machine sont eux-mêmes des objets ou encore la possibilité d'exporter l'objet « réseau Modnet » vers d'autres programmes ou sur Internet.

4.3.2 Classification des objets de Modnet

La première étape dans l'élaboration d'une structure orientée objet consiste à identifier les différents concepts remarquables du domaine d'application. Dans le cas de Modnet, la mise en évidence des objets ne soulève pas de difficulté majeure.

- ☞ ***Le réseau***

Pour commencer, on peut partir du concept de *réseau*, objet de base de l'architecture, qui contient la structure du modèle et les différents paramètres de simulation. Il offre d'une part tous les services de création, de modification et de gestion d'une structure de réseau et d'autre part des services de lancement, d'interruption et de mesure d'état de simulations.

- ☞ ***Les noeuds***

Chaque réseau est composé d'une série de noeuds et de lignes reliant ces noeuds. On peut donc déjà mettre en avant un objet générique *noeud* duquel sont dérivés les objets *institut* et *nuage*, comme le montre le Schéma 7 ci-dessous. Cette

généralisation rendra plus faciles d'éventuelles tentatives d'ajout de nouveaux types de noeuds.

Les instituts seront essentiellement en charge de création, de la gestion et de la validation de leur programme de génération de tâches tandis que le rôle des nuages sera de simuler le délai encouru par le passage des données au travers d'un sous-réseau.

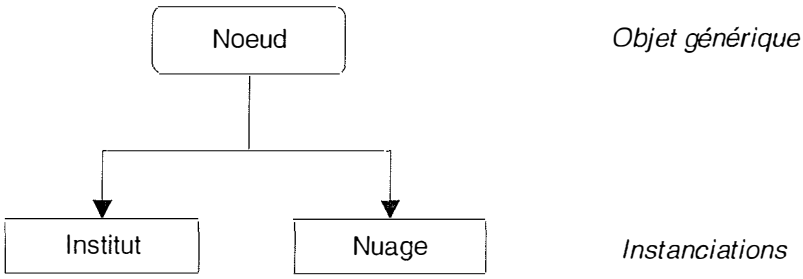


Schéma 7 : L'objet générique "noeud"

☞ *Les lignes*

Tous ces noeuds pourront être reliés par des *lignes*, autre objet remarquable de l'architecture, qui offriront des services de réservation et de libération de bande passante ainsi que des fonctions de simulation de « *background traffic* ». Ce type de ligne est unidirectionnel ; cela signifie que toute connexion physique entre deux instituts se matérialise par une paire d'objets « ligne », chaque objet étant chargé d'un sens du trafic. En cas de connexion unidirectionnelle, la largeur de bande de la ligne non utilisée devra être fixée à 0.

☞ *Les étapes*

Comme dans la précédente version du simulateur, les tâches sont décomposées en *étapes*. Ce nouvel objet décrit l'étape d'une tâche, à savoir : son type et sa taille mesurée en utilisant une mesure de base ou créée par l'utilisateur. Afin d'assurer, une fois de plus, la réutilisabilité des objets de l'architecture, un type d'étape générique a été mis en évidence. Les différentes étapes possibles en découlent comme le montre le Schéma 8. Les principaux services génériques spécifiés dans l'objet « racine » sont des services d'identification de type d'étape et de calcul de taille.

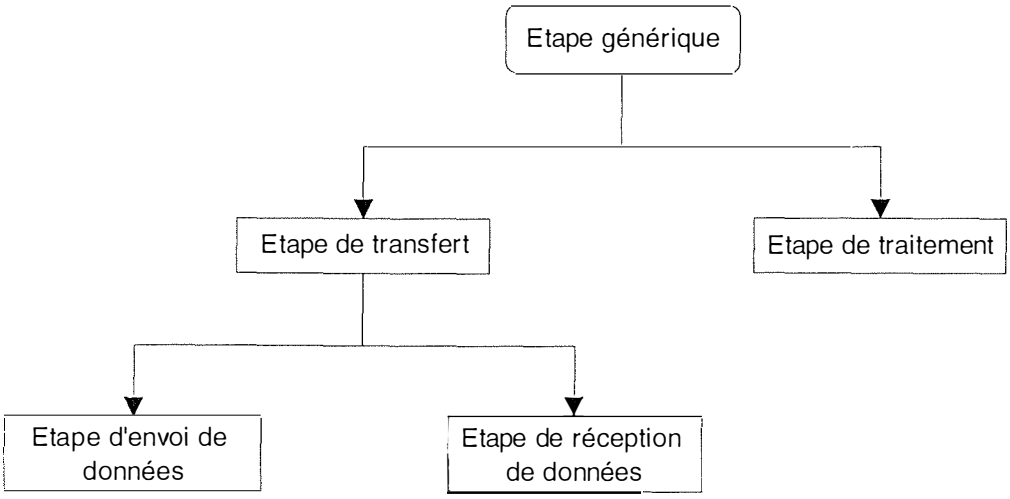


Schéma 8 : Architecture des étapes

☞ *Les unités*

Utilisées lors de l'expression de la taille d'une étape, les *unités* sont des objets simples mais fort utiles qui permettent à l'utilisateur de définir de nouvelles unités à partir d'anciennes. Leurs principaux services sont la validation et la conversion en une unité de base.

☞ *Les tâches*

Avant toute chose, il convient de bien distinguer trois grandes familles d'objets parmi les tâches : les types de tâches (description d'une tâche), les tâches planifiées (tâche programmée dans un institut) et les tâches générées (tâches générées puis exécutées à un institut).

Une première solution aurait été de mettre le type de tâche en tête de la hiérarchie orientée objet, comme proposé au Schéma 9. De cet objet sont dérivées les tâches planifiées et les tâches générées. Une autre solution serait de dériver les tâches générées des tâches planifiées, comme proposé au Schéma 10. Sémantiquement parlant, cette seconde proposition est la meilleure car une tâche planifiée est un type de tâche qui est utilisé dans un institut particulier tandis qu'une tâche générée est l'instanciation d'une tâche planifiée. Pourtant, toutes les informations et les services propres aux tâches planifiées ne sont d'aucune utilité aux tâches générées ; ces dernières seraient inutilement alourdies alors qu'elles sont abondamment manipulées par le simulateur. La première solution semble donc la plus pragmatique et sera préférée à la seconde.

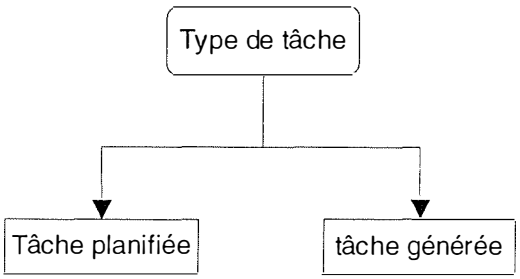


Schéma 9 : Le "type de tâche" au sommet de la hiérarchie (version 1)

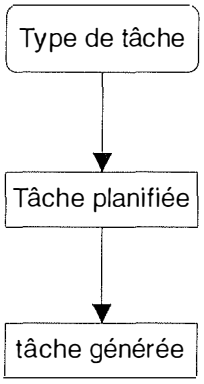


Schéma 10 : Le "type de tâche" au sommet de la hiérarchie (version 2)

Voyons à présent pourquoi l'architecture proposée au Schéma 9 n'est toutefois pas une solution idéale à notre problème. C'est toujours dans un objectif de réutilisabilité que d'autres modifications s'avèrent indispensables.

Le système actuel de génération de tâches est en effet totalement inadéquat pour assurer la possibilité de spécifier de nouveaux types d'étapes. Ainsi, les tâches générées exécutent statiquement les différentes étapes qui composent le type de tâche correspondant. Par « statiquement », on entend ici un branchement vers une routine de traitement choisie selon le type de l'étape en cours.

Une autre optique serait de générer des tâches qui ne s'occupent que de l'exécution d'une étape. A la fin de l'exécution, s'il reste d'autres étapes à simuler, une autre tâche, munie de routines de mises à jour adéquates, est générée pour traiter l'étape suivante.

Bref, les instituts ne généreront plus de tâches mais des étapes qui se passeront successivement les informations relatives au déroulement de la tâche. Notons que, dans un souci de clarté, nous parlerons toujours de « tâche générée » bien que la génération ait été décomposée en étapes. Le Schéma 11 décrit simplement cette modification de stratégie.

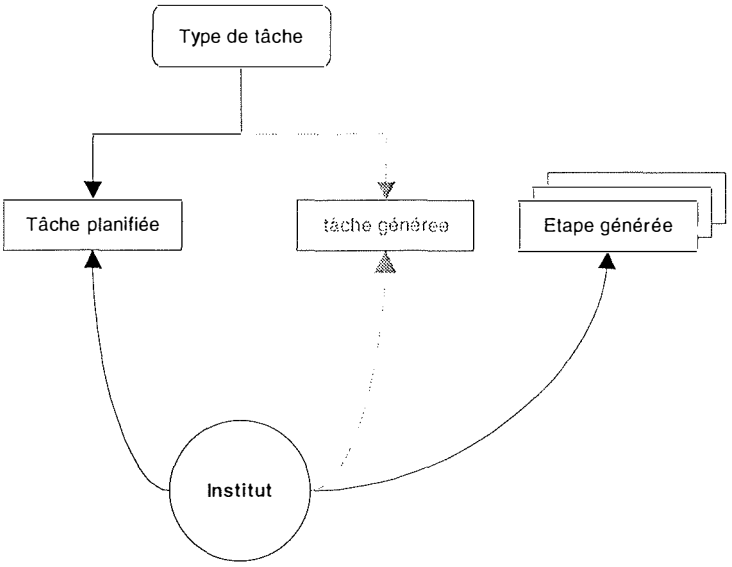


Schéma 11 : Nouveau mécanisme de génération de tâche

Le type de chaque tâche générée pouvant varier selon le type de l'étape correspondante, il importe de faire suivre à l'architecture des tâches la même structure que celle des étapes afin que chaque tâche ait une définition de l'étape qu'elle doit simuler. Le Schéma 12 présente l'architecture des tâches générées ; on remarque que celle-ci est effectivement organisée comme celle du Schéma 8. Deux nouveaux types de tâches sont toutefois propres à cette architecture : les tâches d'envoi et de réception de données au travers des noeuds « nuage ». Ces tâches sont de simples tâches de transfert de données qui « tournent » dans des noeuds de type « nuage »[8].

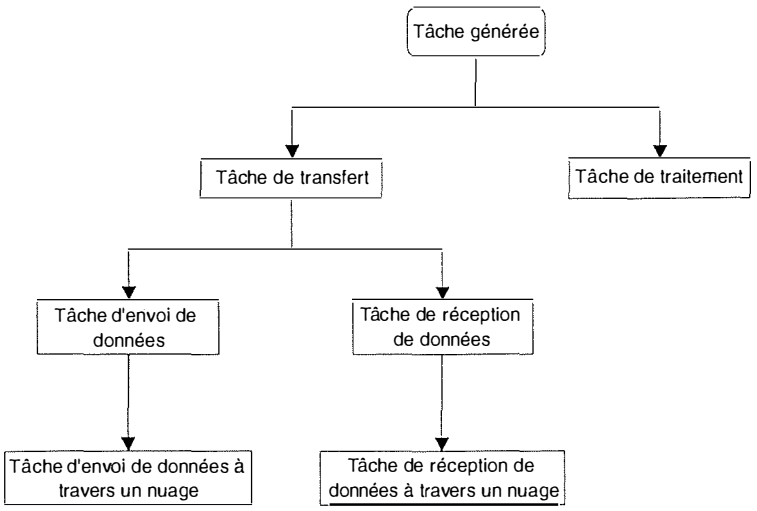


Schéma 12 : Architecture des tâches d'étapes dans Modnet

[8] Se rapporter aux sections 4.5.5 et 4.5.6 pour plus de détails sur cette situation.

4.3.3 Les listes

De nombreux objets de Modnet possèdent parmi leurs propriétés une ou plusieurs listes. Par exemple, le réseau possède une liste de noeuds et les types de tâches possèdent une liste d'étapes. Toutes ces listes ont un grand nombre de services en commun ; généraliser le concept de liste pour obtenir un objet générique devient alors une bonne solution pour éviter le recopiage de code, les problèmes de maintenance et les coûts supplémentaires de programmation.

Ainsi, un objet générique « Liste abstraite » a été implémenté, les éléments de cette liste étant eux-mêmes des objets desquels seront dérivés d'autres éléments. Les schémas 13 et 14 montrent une partie de l'architecture de ces objets avec leur liste correspondante.

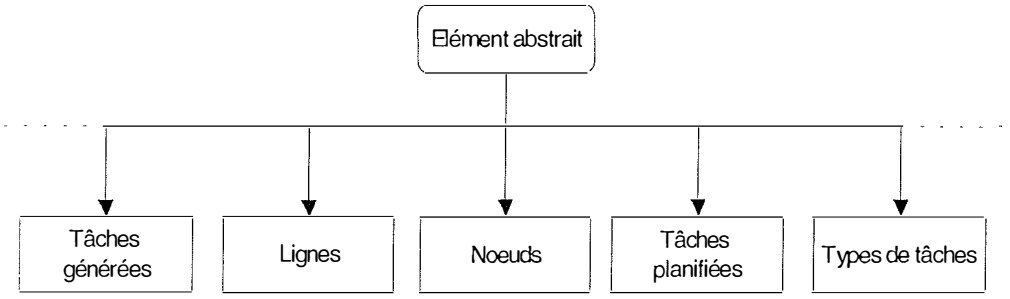


Schéma 13 :L'élément générique et ses dérivés

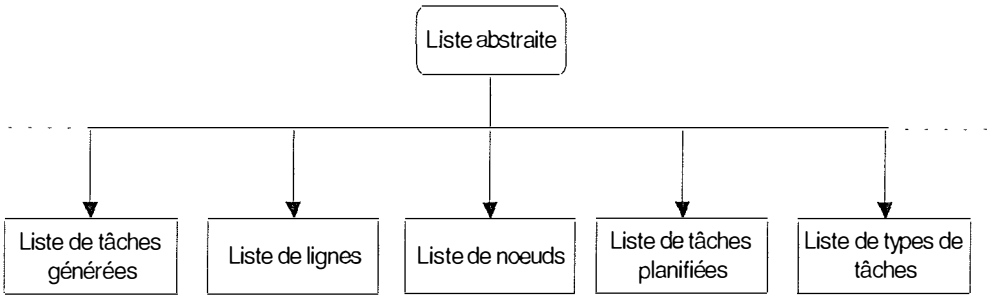


Schéma 14 : La liste abstraite et ses dérivés

4.3.4 L'architecture « Document - vue »

L'architecture « Document - vue » est le noeud central de la plupart des architectures d'applications des logiciels à interface graphique évoluée. Elle s'inspire quelque peu des objets *Model/View/Controller* du monde Smalltalk.

Pour parler simplement, l'architecture « Document - vue » sépare les données de la vision que l'utilisateur en a. Un avantage évident est qu'on peut gérer de multiples vues sur les mêmes données. Par exemple, un document stocké sur disque qui contient les cotations boursières du mois pourrait disposer d'une vue sous forme de table et d'une autre sous forme graphique (Cfr. Schéma 15). Si l'utilisateur venait à modifier les valeurs dans la fenêtre affichant la vue *table*, la fenêtre contenant la vue graphique changerait automatiquement puisque les deux fenêtres affichent en fait les mêmes informations (représentées différemment).

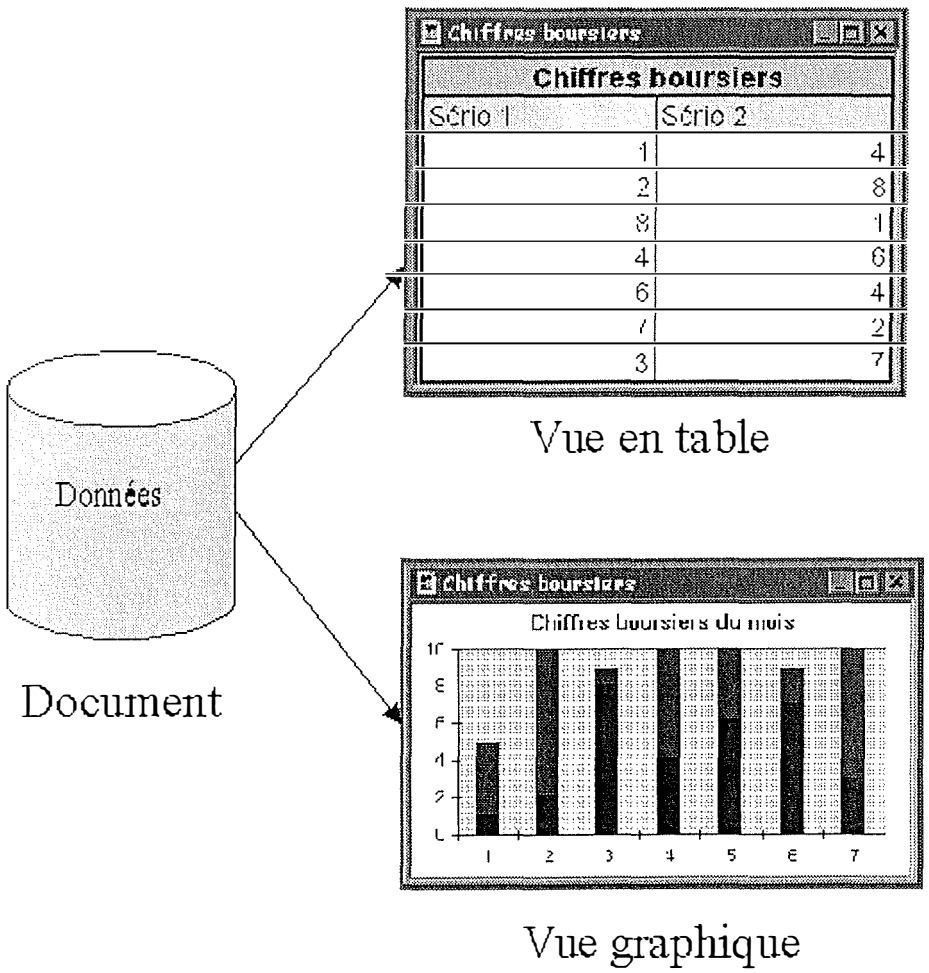


Schéma 15 : Exemple de vues d'un document unique

Modnet, qui est construit sur l'architecture « Document - vue », ne possède pour l'instant qu'une seule vue, à savoir la représentation du réseau sous forme graphique (cfr. section 4.4.2). Mais rien n'empêche le programmeur d'en créer d'autres et de les inclure dans l'architecture du logiciel. Par exemple, au lieu de visualiser le réseau comme un ensemble de lignes et de noeuds, on peut très bien imaginer un tableau dont chaque ligne reprend les statistiques de simulation d'un institut (nombre de tâches en cours d'exécution, ressources utilisées, etc.).

Le document géré par Modnet contient simplement le réseau (lignes et noeuds) ainsi que la liste des différents types de tâches qui peuvent y être simulées.

4.3.5 La découpe en modules

Vu la structure OO adoptée pour l'architecture de Modnet, le regroupement des fonctions en modules ne pose guère de problèmes : à chaque objet du logiciel correspond un module. Une exception existe toutefois pour les objets de type « liste » (cfr. Schéma 14) qui sont inclus dans le module du type d'objet qu'ils manipulent.

4.3.6 Une alternative à l'architecture OO ?

Une architecture de type différent aurait très bien pu être construite. Par exemple, on aurait pu suivre le traditionnel regroupement des modules de type fonctionnel. Parmi ces modules, on trouverait certainement un module gérant la génération de tâche, un module gérant la génération de nombres aléatoires, un module de mise à jour de l'état des tâches en cours ou encore un module de lecture et d'écriture des fichiers de données.

4.4 L'interface de Modnet

4.4.1 Introduction

Un des principaux objectifs de cette nouvelle version du simulateur est d'apporter une nouvelle interface qui permette une utilisation plus intuitive, plus interactive et plus efficace du logiciel. Un aperçu de la fenêtre principale est donné à l'Ecran 2 ci-dessous.

La description de l'interface de Modnet sera faite en utilisant la grille des critères ergonomiques de la méthodologie *Trident*[9] [VAND95].

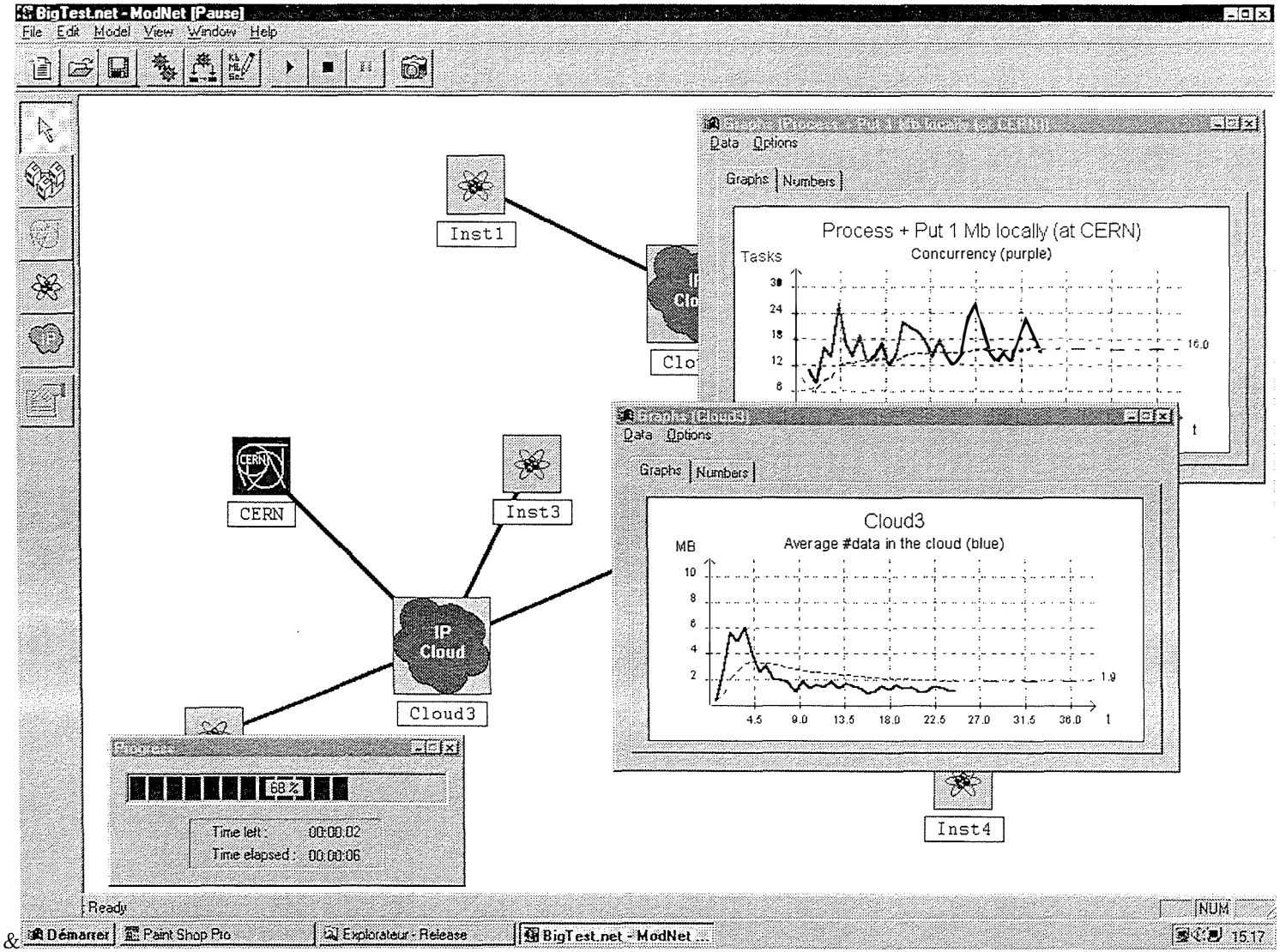
Les différents critères sont :

- La compatibilité
- La cohérence
- La charge de travail
- L'adaptabilité
- Le contrôle de dialogue
- La représentativité
- Le guidage
- La gestion d'erreur

4.4.2 Critère « Compatibilité »

Définition : « *Une interface homme-machine est qualifiée de compatible si et seulement si le (re)codage d'informations et de tâches du monde réel en données et actions du système est réduit. [...] La compatibilité est ici interprétée comme une cohérence avec l'environnement extérieur à l'application (p. ex. les attentes de l'utilisateur, ses habitudes comportementales, les procédures de gestion, les documents sources).* »

[9] *Trident* est une méthodologie de développement d'applications interactives mise au point aux Facultés Universitaires Notre-Dame de la Paix de Namur par une équipe de professeurs et de chercheurs dont les principaux membres sont F. Bodart et J. Vanderdonckt.

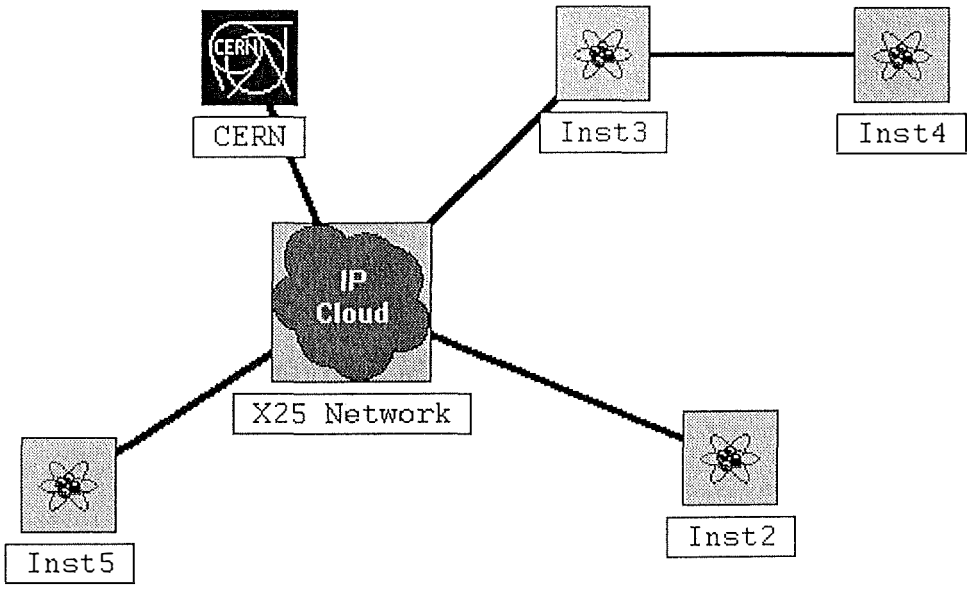


Ecran 2 : Aperçu de la fenêtre principale de Modnet

Le critère de compatibilité transparaît dans Modnet essentiellement au niveau sémantique : la représentation visuelle du réseau modélisé par l'utilisateur correspond à la représentation mentale traditionnelle d'un réseau (ensemble de noeuds reliés par des lignes). Pour favoriser cette compatibilité, chaque type de noeud peut posséder sa propre icône, qui peut éventuellement être définie par l'utilisateur pour favoriser la représentation qu'il se fait de son modèle. L'Ecran 3 montre un exemple de réseau tel qu'il apparaît dans la fenêtre principale de Modnet.

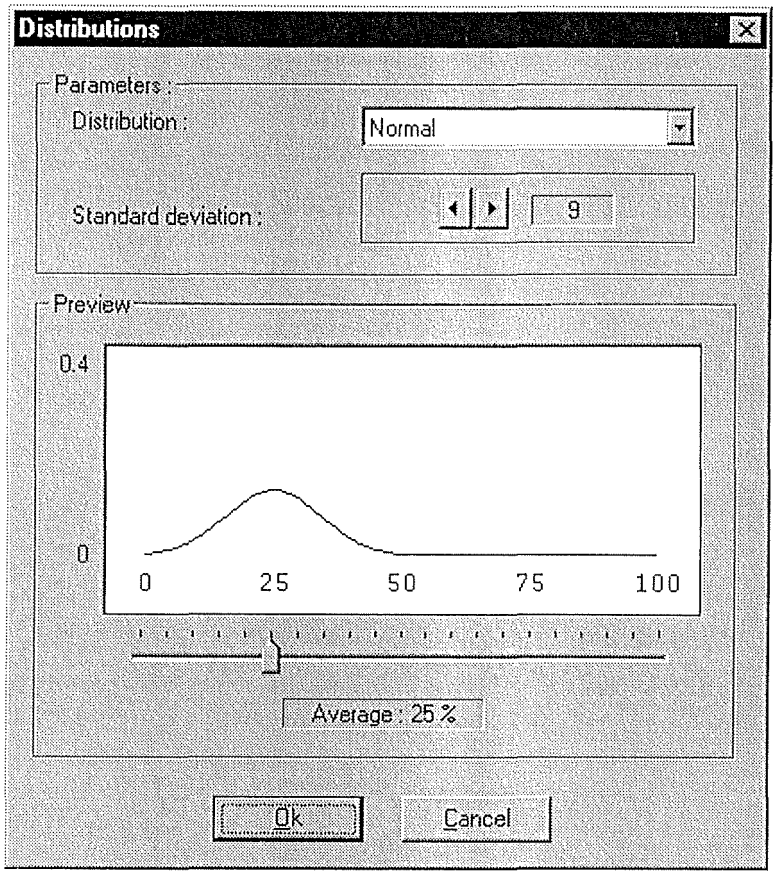
Le même effort de compatibilité a été réalisé lors de la construction de la fenêtre de spécification d'une distribution statistique. L'Ecran 4 montre que l'utilisateur dispose toujours d'une traduction graphique de la distribution qu'il est en train de spécifier.

La compatibilité sémantique se manifeste aussi au niveau de certaines actions que peut effectuer l'utilisateur. Par exemple, les noeuds peuvent être déplacés par simple « glisser - déposer » (*drag and drop*) tel que l'utilisateur le ferait s'il pouvait manipuler les noeuds lui-même.



Ecran 3 : Exemple de réseau construit avec Modnet

Enfin, une certaine compatibilité syntaxique existe dans la méthode d'élaboration d'un projet de simulation : modélisation du réseau, spécification des propriétés, simulation proprement dite, interprétation des résultats.



Ecran 4 : La fenêtre de spécification d'une distribution statistique

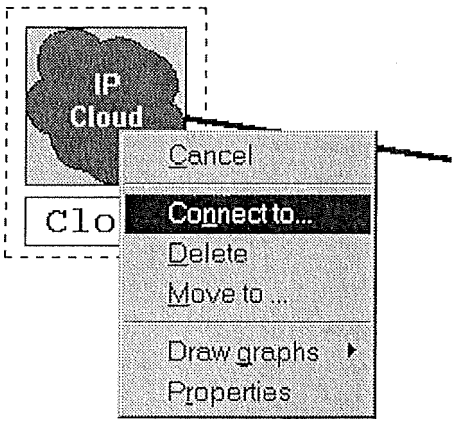
4.4.3 Critère « Cohérence »

Définition : « *Un interface homme-machine est qualifiée de cohérente si et seulement si les données et les actions sont facilement identifiables, reconnaissables et utilisables. En effet, les données sont d'autant mieux perçues et les actions d'autant mieux accomplies qu'elles sont présentées de manière stable et uniformisée. Ce critère concerne la cohérence d'une application avec d'autres applications ou elle-même.* »

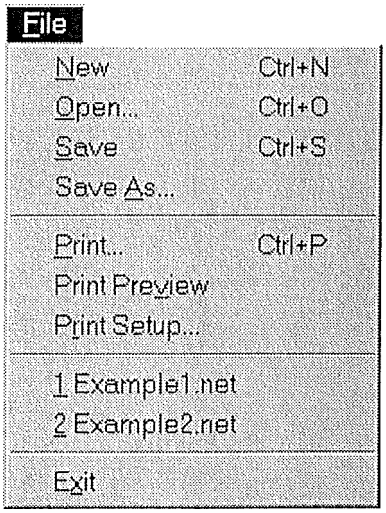
Comme on le dit dans la définition ci-dessus, la cohérence peut être mise en évidence au sein de l'application ou entre applications.

Modnet a été conçu pour s'intégrer de manière cohérente dans son environnement et dans les standards qui y sont couramment utilisés ; ceci favorise la correspondance entre les attentes de l'utilisateur habitué aux commandes de Windows 95 et les séquences d'actions à effectuer dans Modnet. En effet, Modnet reprend plusieurs particularités de cet environnement, comme par exemple :

- Raccourcis-clavier des commandes communes : accès aux menus déroulants (Cfr. Ecran 5), à la fenêtre d'aide, lancement de l'impression, ouverture d'un fichier, etc.
- Utilisation du bouton droit de la souris pour effectuer les actions principales sur les objets du réseau ou éditer leurs propriétés (Cfr. Ecran 6).
- Utilisation de barres d'outils amovibles contenant les fonctionnalités les plus couramment utilisées.
- Utilisation de « tool tips[10] » et de la barre d'état pour donner un commentaire de chaque commande d'un menu déroulant ou d'un bouton d'une barre d'outils.
- Livraison du logiciel sous forme de programme d'installation entièrement compatible avec Windows 95.



Ecran 6 : Menu contextuel en cliquant sur le bouton droit de la souris



Ecran 5 : Menu "File" muni de raccourcis-clavier standards

Au sein de l'application elle-même, l'architecture orientée objet implique une certaine cohérence dans les noeuds du réseau. Tous partagent un certain nombre de propriétés éditables de la même manière. De même, les concepts de type de tâche, de tâche planifiée et de tâche générée sont intégrés de manière transparente et se confondent pour l'utilisateur.

[10] Les « tool tips » sont des petites étiquettes qui apparaissent au-dessus d'un objet lorsque l'utilisateur y laisse le pointeur de sa souris pendant un temps de l'ordre de la seconde. Ces étiquettes contiennent généralement un bref descriptif de l'objet.

Au niveau procédural, une certaine cohérence est maintenue lors de l'exécution de certaines actions comme, par exemple, l'édition des propriétés des objets du réseau (click avec le bouton droit de la souris puis sélection de l'item « propriétés »).

Enfin, la cohérence est restée un critère important dans l'attribution des raccourcis-clavier, dans la disposition et le choix des objets interactifs ou encore dans les libellés et les icônes des boutons de commandes.

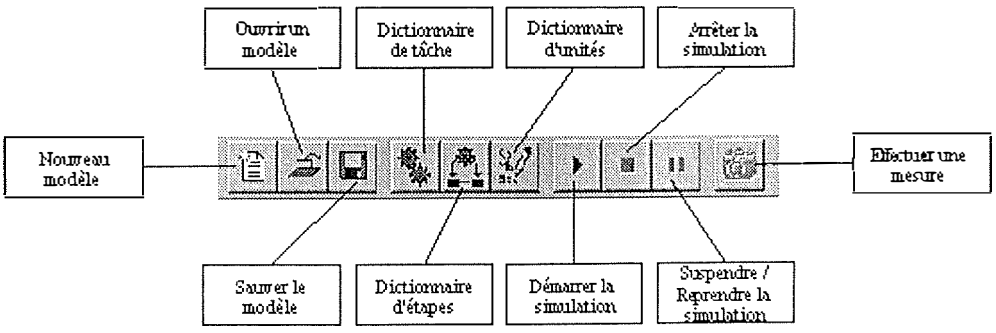
4.4.4 Critère « Charge de travail »

Définition : « *Une interface homme-machine est qualifiée d'efficace en charge de travail si et seulement si le volume de données à manipuler et d'actions à accomplir par unité de tâche est réduit. [...]* »

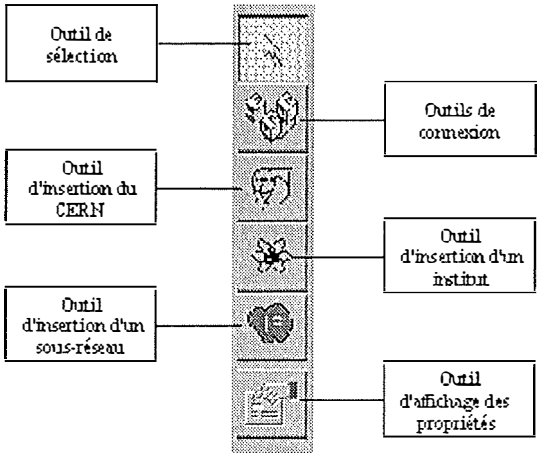
Modéliser un réseau comprenant un grand nombre d'instituts et de lignes peut vite devenir une tâche ardue et monotone. En effet, ne connaissant encore que très peu de choses sur les technologies du moment et leur taux de diffusion au sein des différents instituts, l'utilisateur est bien souvent amené à spécifier les mêmes propriétés pour la majorité des éléments du modèle.

Ainsi, dans un souci d'efficacité en charge de travail, tout nouveau noeud ou toute nouvelle ligne ajoutée dans le modèle prend les propriétés du dernier objet du même type qui y a été inséré.

Une réduction plus classique de la charge de travail se manifeste également dans l'utilisation de deux barres d'outils (Cfr. Ecran 7 et Ecran 8) qui regroupent les actions les plus souvent exécutées par l'utilisateur.



Ecran 7 : La barre d'outils principale



Ecran 8 : Barre d'outils de modélisation

4.4.5 Critère « Adaptabilité »

Définition : « *Une interface homme-machine est qualifiée d'adaptable si et seulement si elle possède la faculté de mimétisme comportemental vis-à-vis de son utilisateur. En effet, l'utilisateur est d'autant moins dérouté et acquerra d'autant plus d'expérience que l'interface peut s'adapter aux différents contextes de travail.* »

L'interface de Modnet est peu adaptable : elle ne s'adapte ni au niveau d'expérience de l'utilisateur ni au contexte de travail.

Il est difficile de différencier l'interface selon l'expérience de l'utilisateur au regard de la simplicité du programme par rapport à la complexité réelle du problème et de l'expérience déjà importante de la majorité des utilisateurs potentiels.

4.4.6 Critère « Contrôle de dialogue »

Définition : « *Une interface homme-machine est qualifiée d'interface à contrôle explicite si et seulement si elle peut fournir à l'utilisateur l'illusion qu'elle est placée sous son contrôle et que ses actions s'exécutent suite à des demandes explicitement formulées par l'utilisateur. Une interface est qualifiée d'interface à contrôle implicite si et seulement si elle est placée sous le contrôle du système. Elle est qualifiée*

d'interface à contrôle mixte lorsqu'elle est tour à tour à contrôle explicite et implicite. »

L'interface de Modnet est sans conteste à contrôle mixte. Elle est en effet successivement à contrôle explicite et implicite.

L'interface est à contrôle explicite lors de toute la phase de modélisation. L'utilisateur place lui-même les différents objets et définit lui-même les propriétés de chaque élément du modèle. Un certain contrôle implicite existe néanmoins dans la récupération par le système des propriétés du dernier objet inséré lors de l'ajout d'un nouveau noeud ou d'une nouvelle ligne.

L'interface est à contrôle implicite lors de la phase de simulation. C'est en effet le programme qui détermine quand de nouvelles tâches doivent être générées grâce à son générateur de nombres aléatoires. C'est également lui qui traite et qui formate les résultats sous forme de graphiques ou qui les écrit dans un fichier de sortie. Un certain contrôle explicite persiste pourtant dans la phase de simulation. En effet, le système vérifie régulièrement si l'utilisateur n'essaie pas d'exécuter une action autorisée (arrêter la simulation, effectuer une mesure, etc.) ou de manipuler les objets interactifs du logiciel (déplacement ou réduction des fenêtres, utilisation des menus, etc.). L'utilisateur garde ainsi la possibilité d'intervenir dans le processus de simulation malgré que celle-ci soit pilotée par le système.

4.4.7 Critère « Représentativité »

Définition : « *Une interface homme-machine est qualifiée de représentative si et seulement si les codes utilisés, les items de menus, les libellés facilitent l'encodage et la rétention.* »

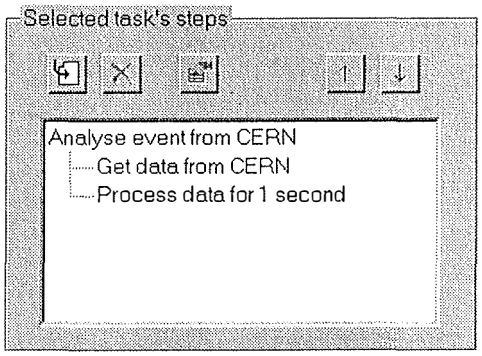
La représentativité est un critère important dans la conception de l'interface de Modnet. Chaque bouton, chaque libellé et chaque raccourci-clavier a été choisi de manière à favoriser la rétention de sa signification ou de son moyen de déclenchement.

Ainsi, une attention particulière a été portée au choix de l'icône qui illustre chaque bouton de commande sans libellé, à savoir les boutons des barres d'outils ou certains boutons des boîtes de dialogue. Par exemple, les boutons de gestion de la simulation (démarrage, arrêt, pause) représentent les traditionnelles commandes d'un lecteur de médias, comme on peut le voir sur l'Ecran 7 ci-dessus.

Sur ce même Ecran 7, les boutons de gestion du fichier (nouveau modèle, chargement et sauvegarde) possèdent l'icône standard des applications Windows 95 qui sont d'une bonne représentativité ; ceci renforce également la cohérence inter-applications.

Dans le but de favoriser la rétention des raccourcis-clavier, la priorité a été donnée à la première lettre du libellé de la commande. Le défi est relevé pour pas moins de 75% des raccourcis-clavier tout en conservant les raccourcis standards comme ceux du menu « File ».

Pour terminer cette section relative au critère de représentativité, notons qu'une représentation explicite de la décomposition d'une tâche en étapes a été utilisée dans la boîte de dialogue d'édition des types de tâche. L'Ecran 9 ci-dessous montre en effet que l'utilisation d'une structure hiérarchique visuelle facilite la rétention de ce concept de décomposition.



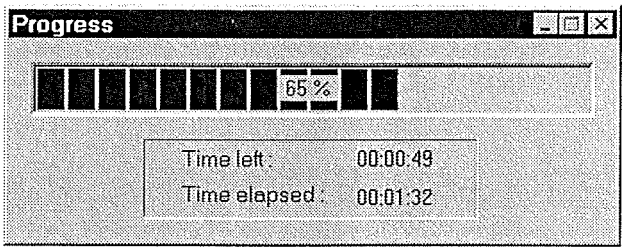
Ecran 9 : Décomposition visuelle d'une tâche en étapes

4.4.8 Critère « Guidage »

Définition : « *Une interface homme-machine est qualifiée d'efficace en guidage si et seulement si elle informe de manière constante l'utilisateur de l'issue de ses actions et sur sa position dans l'accomplissement de sa tâche. [...]* »

Une donnée critique pour l'utilisateur lors du déroulement d'une simulation est le temps qu'il reste avant que celle-ci soit terminée. Le moyen le plus efficace et le

plus visuel est l'utilisation d'une barre de progression mise à jour en temps réel lors de la simulation ; cet objet interactif étant également un excellent argument dans l'évaluation du critère de représentativité. L'Ecran 10 montre que le temps écoulé et une évaluation du temps restant font également partie de la fenêtre de progression.

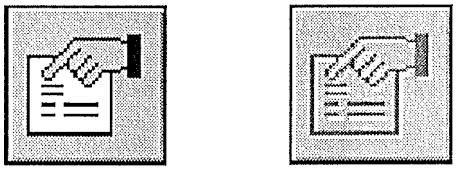


Ecran 10 : La barre de progression de la simulation

Toujours lors de la simulation, l'utilisateur est directement informé de l'état du processus par deux moyens : il peut consulter d'une part l'état des boutons de la barre d'outils et, d'autre part, remarquer un message supplémentaire (« [Run] » ou « [Pause]) dans la barre de titre à côté du nom du fichier, comme montré à l'Ecran 11.

Ecran 11 : Etat de la simulation dans la barre de titre

Outre le fait que l'utilisateur connaît en permanence l'état d'avancement de la modélisation en ayant constamment la représentation graphique du réseau sous les yeux, le programme lui indique toujours si les différentes actions disponibles peuvent être effectuées ou non. Cette information est visuellement donnée par la méthode classique de l'activation / désactivation. Sous Windows 95, la désactivation se manifeste par le « grisage » de l'objet concerné, comme le montre l'exemple de l'Ecran 12 dans le cas du bouton de commande. Tout objet « grisé » ne peut être manipulé par l'utilisateur et ce jusqu'à ce qu'il soit réactivé.



Ecran 12 : Désactivation du bouton d'édition de propriétés

Enfin, l'interface de Modnet peut également être qualifiée d'efficace en termes de guidage grâce à sa série de curseurs de souris qui informent l'utilisateur de l'action susceptible d'être effectuée lors de son prochain « click » dans le modèle. Le Tableau 5 ci-dessous montre la liste de ces curseurs.

Tableau 5 : Les curseurs souris de Modnet

Curseur	Action(s) possible(s)
	Sélection d'une ligne ou d'un noeud
	Sélection du premier noeud lors de l'ajout d'une ligne.
	Sélection du second noeud lors de l'ajout d'une ligne.
	Insertion d'un nouvel institut (qui n'est pas le CERN)
	Insertion du CERN (s'il n'est pas encore dans le modèle)
	Insertion d'un sous-réseau
	Déplacement d'un noeud.

4.4.9 Critère « Gestion d'erreurs »

Définition : « *Une interface homme-machine est qualifiée d'efficace en gestion des erreurs si et seulement si elle s'avère robuste vis-à-vis des erreurs commises par l'utilisateur et se montre conviviale dans la manière de les corriger.* »

Une simulation pouvant parfois durer très longtemps, il est nécessaire de s'assurer que tous les paramètres spécifiés par l'utilisateur sont corrects et qu'il n'existe pas d'incohérences dans le réseau. Ainsi, chaque paramètre manquant ou incorrect est immédiatement refusé et un message d'erreur explicite informe l'utilisateur de son erreur. Quand celui-ci en a pris note, le champ erroné est directement éditable sans aucune utilisation de la souris.

Lorsque la simulation est lancée, une vérification de l'intégrité du réseau et des tâches est effectuée. Parmi les erreurs recherchées, on vérifie par exemple que chaque tâche possède au moins une étape, qu'il existe des tâches à générer, que tous les noeuds sont accessibles, etc. Le système vérifie également si la largeur de bande spécifiée pour chaque ligne est suffisamment grande pour supporter le trafic prévu (calculé approximativement en se basant sur les paramètres donnés). En cas de trafic trop important, l'utilisateur en est informé mais la simulation peut néanmoins commencer.

4.5 Principes & fonctionnement

L'objectif de cette section est d'expliquer les différents concepts de Modnet, qu'ils soient nouveaux ou repris de l'ancienne version, ainsi que les méthodes utilisées pour effectuer la simulation.

4.5.1 Le réseau et ses composants

☞ ***Les instituts***

Le concept d'institut dans l'environnement de Modnet est le même que dans la précédente version du simulateur, à savoir une institution qui désire participer activement ou passivement au partage des données relatives aux expériences LHC.

Ces instituts possèdent toujours la possibilité de générer un certain nombre de tâches selon une certaine fréquence dont la distribution statistique peut être uniforme, normale ou de Poisson. En outre, chacun disposera d'une quantité de ressources qui lui est propre (quantité et puissance des processeurs, largeur de bande du réseau local, données stockées sur disque) et connaîtra le prix de chaque nouvelle ressource achetée.

Le CERN est un institut particulier : une icône spéciale le caractérise afin qu'il soit plus facilement identifiable dans le modèle. L'Ecran 13 ci-dessous montre la différence visuelle entre ces deux types d'instituts.

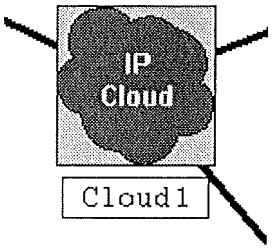
Ecran 13 : Les icônes du CERN et des autres instituts

☞ *Les sous-réseaux*

Les différents instituts qui composent le réseau sont rarement tous reliés par des lignes louées ; bien souvent, les connections passent par des sous-réseaux publics ou privés. Or, connecter directement deux institutions avec une ligne louée ou via une connexion de part et d'autre d'un sous-réseau n'implique pas les mêmes performances en terme de temps de transfert ou de coûts. Modnet n'a certainement pas la prétention de simuler avec précision le comportement d'un sous-réseau, tel n'est pas son objectif. Cette notion a été essentiellement introduite dans le modèle pour simuler le délai qu'entraîne le passage des données par un sous-réseau.

Ainsi, pour simuler le délai de passage, Modnet a besoin de savoir le temps moyen que prend un paquet de données d'une taille déterminée pour traverser le sous-réseau, les connexions nécessaires étant supposées ouvertes. En ce qui concerne les coûts, le prix du transfert d'un paquet de taille donnée doit également être spécifié par l'utilisateur.

Dans le modèle, un sous-réseau peut être identifié par l'icône présentée à l'Ecran 14.



Ecran 14 : L'icône d'un sous-réseau

☞ *Les lignes*

Comme introduit à la section 4.2.2 relative aux nouveautés de Modnet, la notion de ligne est à présent plus explicite et existe en dehors de tout transfert de données planifié. Les lignes doivent être construites par l'utilisateur lors de la spécification du modèle.

Bien que chaque ligne manipulée par le simulateur soit unidirectionnelle, le fait de connecter deux noeuds du réseau crée automatiquement une paire de lignes (une dans chaque sens), chacune d'entre elles pouvant bénéficier de ses propres propriétés.

Bien sûr, ces lignes n'ont pas été mises en place pour supporter exclusivement le trafic engendré par les transferts de données relatives aux expériences du LHC et d'autres données y circuleront très certainement de manière continue. Ainsi, l'utilisateur peut spécifier, pour chaque ligne, un trafic secondaire (ou d'arrière-plan) qui utilise une certaine portion de la largeur de bande et qui peut varier en suivant une distribution statistique normale, constante ou exponentielle.

Enfin, l'utilisateur peut déterminer la largeur de bande maximale pouvant être affectée aux transferts de données ; la largeur de bande restante est utilisée par la signalisation ou les paquets retransférés suite à des erreurs de transmission. Par exemple, dans le protocole ATM, le trafic qui passe sur une ligne ne peut dépasser 70% de la largeur de bande, le reste étant réservé aux nombreux canaux privés réservés notamment à la signalisation.

4.5.2 Les tâches

Chaque institut peut donc planifier un certain nombre de tâches qui seront la source de toute la simulation. Elles permettront d'observer dans quelle mesure elles affectent l'utilisation de chaque type de ressource (processeurs, espace disque, largeur de bande). La section 4.3.2 explique en détail la notion de tâche dans Modnet ainsi que la décomposition en étapes.

4.5.3 Les unités

Modnet permet à l'utilisateur de définir ses propres unités pour améliorer la sémantique de la spécification d'une tâche. Chaque nouvelle unité est simplement un multiple d'une unité existante. Grâce à ce nouveau concept, l'utilisateur est en mesure d'enrichir la spécification d'une tâche en la définissant au moyen d'unités plus en rapport avec le domaine d'application. Le Schéma 16 ci-dessous montre un exemple fictif de définition d'unités par l'utilisateur. Chaque cadre représente une unité, le cadre dont le libellé est en gras est une unité de base, toujours disponible dans le programme.

Les arcs représentent, quant à eux, le multiple qui distingue toute nouvelle unité de son unité de référence. Par exemple, une *description photo de collision* est un ensemble de 8 *photos de collision* dont chacune est égale à 850 *Kilo-octets*. L'utilisateur pourra ainsi spécifier plus clairement une tâche « *traitement et stockage de rapports de température* » (cfr. Schéma 17).

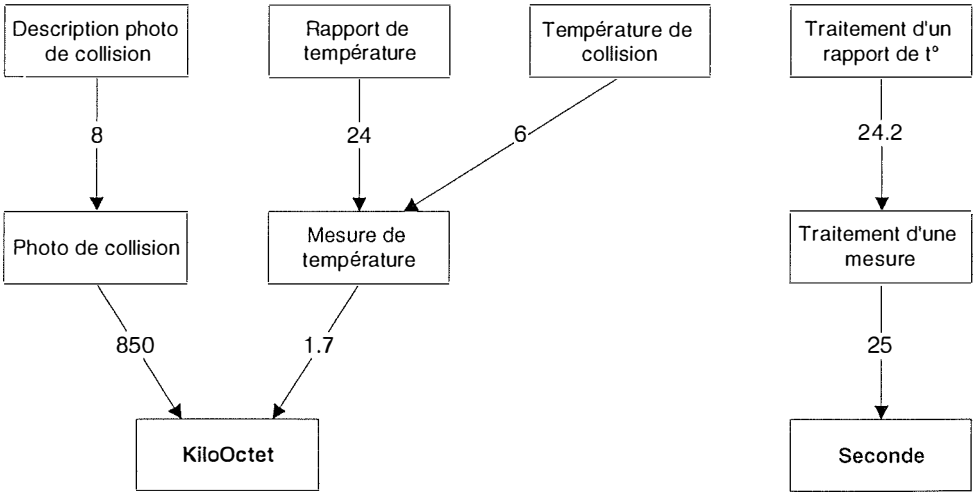


Schéma 16 : Exemple d'unités définies par l'utilisateur

Vu qu'il est également important de connaître la taille réelle de chaque étape d'une tâche, Modnet parcourt l'arbre de définition des unités jusqu'à arriver à une unité de base afin de donner à l'utilisateur cette information importante. Par exemple, une *description photo de collision* occupe une place de 8 x 850 = 6800 Ko.

Pour s'assurer que ce processus se déroule sans problème, il faut empêcher l'apparition de cycles dans la définition des unités. La présence de cycles est ainsi recherchée lors de chaque nouvelle déclaration d'unité pour éviter ce problème.

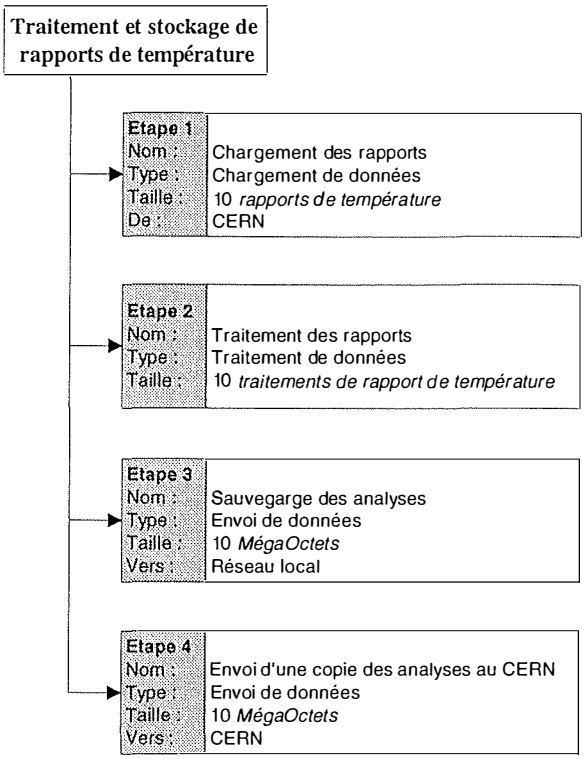


Schéma 17 : Exemple de tâche utilisant les unités du schéma précédent

4.5.4 La génération de tâches

Le mécanisme de génération de tâches de Modnet ressemble fort à celui de la version précédente du simulateur. En effet, à chaque tranche de temps, pour chaque tâche planifiée de chaque institut, un nombre aléatoire est produit grâce au générateur de nombres aléatoires de distribution uniforme (dont la validité a été étudiée au chapitre 3). Selon la fréquence spécifiée par l'utilisateur et la distribution statistique associée (seule une distribution uniforme était utilisable dans le premier simulateur), une tâche est éventuellement générée et initialisée.

4.5.5 La réservation des ressources

Lors de la création de chaque tâche, une demande de réservation de ressources est lancée. Selon le type de la tâche, un processeur ou une partie de la largeur de bande d'une ligne sera requise pour l'exécution.

☞ *La ressource « processeur »*

Pour les tâches de traitement, rappelons qu'un processeur est réservé à 100 % pour toute nouvelle occurrence. Si l'utilisateur a limité le nombre de processeurs de l'institut qui a généré la tâche, la réservation peut éventuellement être refusée et reportée. Dans ce cas, la tâche est suspendue et attend qu'un processeur se libère pour pouvoir s'exécuter. Par contre, si le nombre de processeurs disponibles à l'institut est illimité, un nouveau processeur est alloué et la tâche peut immédiatement commencer.

Lors de leur création, les tâches de traitement se voient attribuer un temps d'exécution inversément proportionnel à la puissance des processeurs de l'institution où elles seront exécutées. Leur taille étant en effet toujours spécifiée en secondes sur un processeur de 100 MIPS, il importe de tenir compte de toute différence de performance du matériel. Bref, si un institut dispose de processeurs de 50 MIPS, la durée du traitement sera deux fois plus longue que celle prévue lors de la définition de la tâche.

☞ *La ressource « réseau local »*

Simuler le déroulement d'une tâche de transfert au sein d'un réseau local ne fait pas partie des objectifs de Modnet. Aucune ressource n'est donc requise pour effectuer ce genre de transfert dont le temps d'exécution est simplement calculé en fonction de la taille des données et de la largeur de bande du réseau local, spécifiée par l'utilisateur lors du paramétrage des instituts.

☞ *La ressource « largeur de bande »*

La ressource-clé des tâches de transfert de données vers un autre institut est sans conteste la largeur de bande des lignes. Celle-ci est toujours partagée équitablement entre les différentes tâches qui l'utilisent. Par conséquent, tout départ ou arrivée d'une tâche implique une redistribution de la largeur de bande à l'ensemble des tâches, un peu à l'instar du concept d'ABR[11] du protocole ATM.

Dans le cas de Modnet, aucune limite n'a été donnée quant au nombre de canaux ouverts. Ainsi, même si beaucoup de tâches requièrent de la largeur de bande, aucune demande ne sera refusée mais la portion allouée sera d'autant plus petite que le nombre de tâches à satisfaire est grand[12].

[11] Adaptative Bit Rate : taux de transfert adaptatif, c'est-à-dire qui dépend de la charge globale de la ligne.
[12] Cette situation peut parfois mener à une situation d'emballement où les tâches s'entassent et ne peuvent se terminer à cause d'une largeur de bande insuffisante.

☞ *La route d'accès d'un noeud à un autre*

Lorsqu'un institut désire effectuer un transfert de données avec un autre institut auquel il n'est pas directement connecté, la réservation de la ressource « largeur de bande » doit être transmise tout au long de la route qui mène à l'institut de destination.

Le choix de la route est évidemment un processus important dans le simulateur car il déterminera quels noeuds et quelles lignes seront utilisés lors des transferts de données. Dans la recherche d'un algorithme adéquat, un double objectif était poursuivi : il fallait d'une part trouver un algorithme peu gourmand en temps de calcul vu le nombre de routes à calculer (chaque institut doit connaître la route vers tous les autres) et d'autre part choisir un critère qui reflète le mieux la réalité.

Le choix du critère de recherche peut être ramené à l'interprétation que l'on peut se faire des arcs (lignes) joignant les différents noeuds du réseau. Trois interprétations se distinguent :

- La distance entre les noeuds
- La largeur de bande de la ligne
- La présence de la ligne

La première solution considère la distance géographique entre les noeuds du réseau comme critère de choix dans la recherche du plus court chemin. Vu la vitesse à laquelle se propagent les données sur les lignes (≈ vitesse de la lumière), la notion de distance a peu d'intérêt. Cette solution est donc à éviter.

La largeur de bande est quant à elle un critère plus intéressant, car elle est souvent déterminante dans le temps de transmission d'une donnée d'un bout à l'autre d'un réseau. Cela pourrait donc être une solution idéale pour trouver la meilleure route. Néanmoins, cette méthode n'a pas été retenue. Tout d'abord, ce critère écarterait trop souvent les lignes peu rapides au profit des « autoroutes ». Ensuite, il s'éloigne assez fort des algorithmes de choix de route des protocoles actuels (suite IP) qui ne connaissent généralement pas la capacité des lignes. Enfin, lors de la modélisation, l'utilisateur spécifie bien souvent les mêmes largeurs de bande pour toutes les lignes ; ce qui nous ramène au troisième choix : la présence de la ligne.

Le dernier critère, la présence de la ligne, consiste simplement à trouver la route qui emprunte le moins de lignes (ou de noeuds) pour arriver à l'institut de

destination. Consistant à attribuer un poids égal à 1 à chaque ligne du réseau, cette méthode est la plus logique et la plus couramment utilisée. Elle sert également de base aux algorithmes de calculs de routes des protocoles de la suite IP, largement répandus de nos jours.

Parmi les algorithmes qui calculent un chemin extrémal dans un graphe, le plus célèbre est sans conteste l'algorithme de *Moore-Dijkstra*. Cet algorithme, peu gourmand en temps de calcul, permet le calcul d'un chemin extrémal (de poids minimal ou maximal) dans un graphe possédant les propriétés suivantes :

(P1) Aucun sommet n'est relié à lui-même (pas de boucles)
(P2) Le poids des arcs est toujours ≥ 0 (si chemin minimal) ou ≤ 0 (si chemin maximal).

La première propriété est assurée par Modnet : aucune ligne ne peut être construite entre un institut et lui-même. De plus, les envois de données vers le réseau local ne sont pas concernés par la recherche d'un chemin.

La seconde propriété est également vérifiée puisque, suite au débat concernant le critère d'évaluation, tous les arcs (lignes) ont été munis d'un poids égal à 1.

L'algorithme de *Moore-Dijkstra* est donc applicable aux réseaux construits avec Modnet.

Cet algorithme, dont le listing et la démonstration se trouvent en annexe, fait partie de la famille des *algorithmes par extension sélective* [FICH95] dont le principe est similaire au raisonnement inductif. En effet, il gère au départ une solution de base simple qu'il étend progressivement jusqu'à parvenir à la solution demandée. Dans le cas de la recherche d'un chemin de poids minimal dans un graphe, le cas de base consiste à trouver un chemin allant du noeud de départ vers lui-même. Le meilleur chemin est alors de ne rien faire (poids nul). A chaque étape, le chemin est calculé vers un sommet de plus en plus éloigné jusqu'à ce qu'on arrive au sommet de destination.

☞ ***Propagation de la réservation de largeur de bande***

Une fois le chemin déterminé, une partie de la largeur de bande des lignes qui le composent doit être réservée pour transférer les données envoyées par la tâche. Cette demande de réservation est propagée de ligne en ligne, en partant du noeud de départ, au fur et à mesure de l'écoulement des tranches de temps. En d'autres termes, chaque réservation de ressources sur la route qui sépare l'institut de départ de l'institut de

destination prend, par convention, une tranche de temps. Par exemple, si au temps T une tâche demande une réservation de largeur de bande entre deux instituts séparés par 4 lignes, les données commenceront à être transmises sur la première ligne au temps $T+4(\Delta t)$.

☞ *Passage d'une ligne à une autre*

Lors du passage d'une ligne à une autre, une tâche dite « de transit » est créée au niveau du noeud intermédiaire pour s'occuper du routage des données vers la ligne suivante. Initialement, ces tâches n'ont aucune donnée à envoyer (contrairement à la tâche du noeud de départ) et attendent d'en recevoir du noeud précédent pour les faire transiter.

Si le noeud intermédiaire est un institut, cette tâche est une simple copie de la tâche de départ.

Si, par contre, le noeud intermédiaire est un sous-réseau, une tâche spéciale y est créée : son rôle sera de retarder les données avant de les transférer vers le noeud suivant afin de simuler le temps qu'elles prennent pour traverser le sous-réseau.

4.5.6 La mise à jour des tâches

A la fin de chaque tranche de temps, toutes les tâches en cours d'exécution du modèle sont modifiées pour tenir compte du temps écoulé depuis la dernière mise à jour.

☞ *Les tâches de traitement*

La mise à jour des tâches de traitement consiste simplement en une incrémentation du temps d'exécution d'une durée égale à la tranche de temps. Ce temps d'exécution augmente ainsi jusqu'à égaler ou dépasser le temps prévu. C'est alors le moment de considérer la tâche comme terminée et de libérer le processeur utilisé.

☞ *Les tâches de transfert de données avec le réseau local*

Aucune interdépendance n'existant entre les tâches de transfert avec le réseau local, la mise à jour se ramène à ajouter à la progression de la tâche la quantité de données transférées pendant la tranche de temps, compte tenu de la largeur de bande du réseau local.

☞ *Les tâches de transfert de données avec d'autres noeuds*

S'il reste des données à transférer, la tâche en envoie une certaine quantité vers le noeud selon la taille de la largeur de bande allouée et la durée de la tranche de temps.

Si la tâche est une tâche « de transit », les données à envoyer viennent du noeud précédent. Si, de plus, cette tâche doit simuler le délai de passage dans un sous-réseau, toute donnée reçue est découpée en paquets dont la taille et la vitesse de transfert sont propres à ce sous-réseau. Lors de chaque mise à jour, on vérifie si des paquets ont terminé de le traverser ; si c'est le cas, les données brutes sont reconstituées et transférées vers le noeud suivant dans la mesure des possibilités de la ligne, sinon le délai associé à chaque paquet est réduit en fonction de la durée de la tranche de temps.

4.5.7 La libération des ressources

Chaque fois qu'une tâche est terminée, elle libère les ressources qu'elle a utilisées lors de son exécution.

☞ *Les tâches de traitement*

Lorsqu'une tâche de traitement est terminée, elle libère le processeur qui, rappelons-le, lui a été affecté à 100% durant toute la durée de l'exécution. Le processeur est ainsi disponible pour d'éventuelles autres tâches de traitement en attente de ressources.

☞ *Les tâches de transfert de données avec d'autres noeuds*

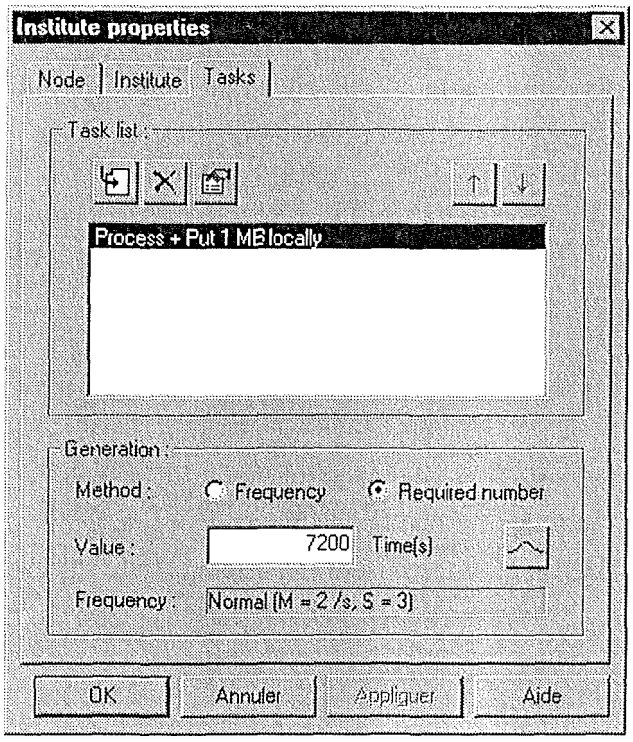
Dès que toutes les données ont été envoyées (ou reçues, selon le type de la tâche), la libération de la largeur de bande réservée est effectuée en commençant par la première ligne. Comme lors de la réservation, la libération des ressources entraîne une redistribution de la largeur de bande parmi les tâches restantes. Ensuite, la demande de libération de ressources est transmise à la ligne suivante qui la recevra à la tranche de temps suivante.

4.6 Exemple de session avec Modnet

4.6.1 Construction du modèle

La construction du modèle consiste donc à placer, à l'aide de la barre d'outils de modélisation, les différents composants qui constituent le réseau, à savoir : les instituts (dont, éventuellement, le CERN), les sous-réseaux et les lignes qui connectent ces différents noeuds entre eux.

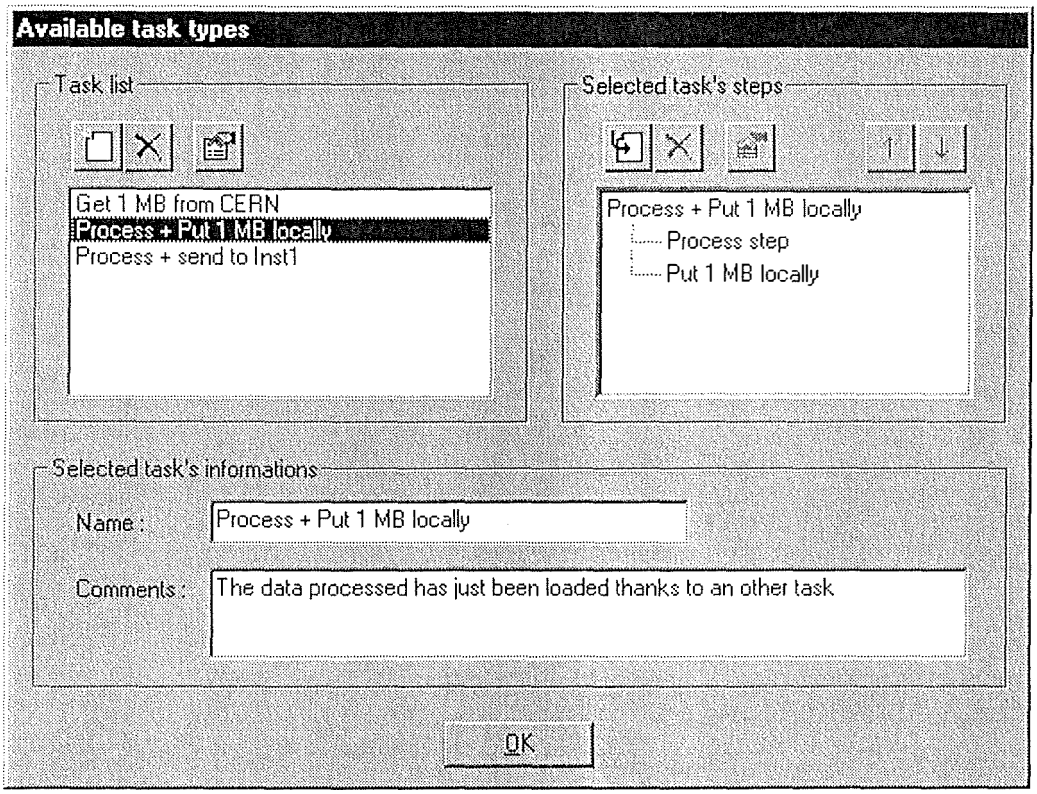
Chaque fois qu'un nouvel objet est ajouté au modèle, Modnet ouvre automatiquement la fenêtre de propriétés correspondante afin que l'utilisateur puisse paramétrer la nouvelle ligne ou le nouveau noeud. Chaque fenêtre de propriétés, comme le montre l'Ecran 15, est constituée d'une série de pages qui regroupent chacune des informations relatives à un certain aspect de l'objet. Par exemple, la feuille de propriétés des instituts contient trois pages : la première rassemble les informations générales concernant l'institut (nom, situation, icône), la deuxième concerne la spécification des ressources (réseau local, processeurs et capacité disque) et de leur coût respectif, la troisième regroupe les informations relatives à la planification des tâches.



Ecran 15 : Fenêtre des propriétés d'un institut

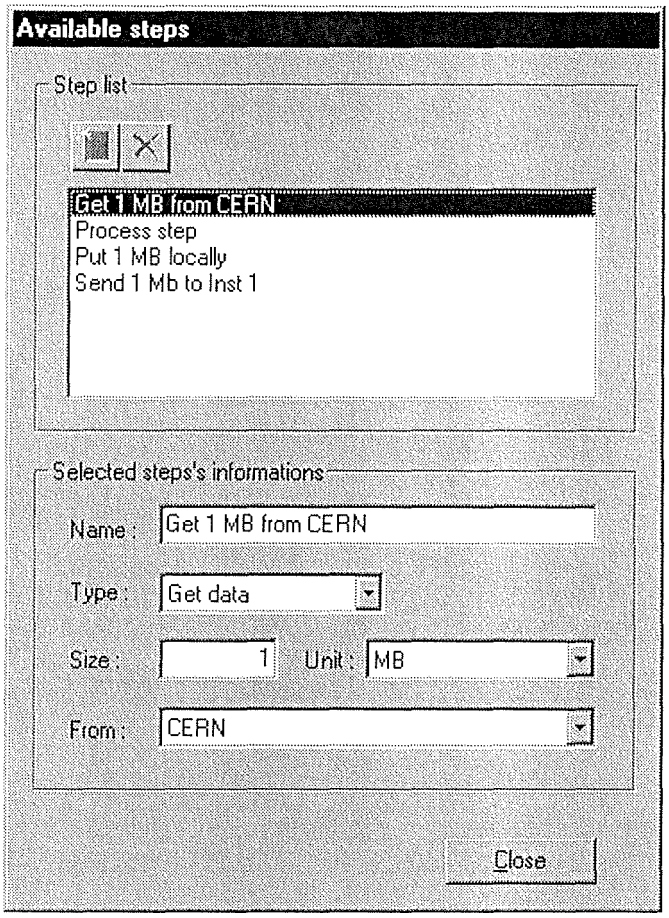
4.6.2 Spécification des tâches

Une fois le réseau construit, il reste à spécifier les différentes tâches qui seront exécutées au sein de ce réseau. Pour ce faire, via la barre d'outils ou les menus déroulants, l'utilisateur dispose d'une fenêtre de construction de tâche, présentée à l'Ecran 16. Grâce aux boutons du groupe de droite de la fenêtre (« Select task's steps »), l'utilisateur peut ajouter, supprimer ou déplacer des étapes.



Ecran 16 : La fenêtre de construction de tâches

Ces étapes sont quant à elles construites grâce à une autre fenêtre (Ecran 17) qui permet la spécification complète d'une étape : nom, type, taille, unité et, selon le type de l'étape, l'entité réseau avec laquelle se fera le transfert de données.



Ecran 17 : Fenêtre de construction des étapes

4.6.3 Planification des tâches

Maintenant que les tâches ont été spécifiées, chaque institut peut décider d'en exécuter un certain nombre. Pour cela, par l'intermédiaire de la fenêtre de propriétés des instituts, l'utilisateur a la possibilité de définir précisément la fréquence et la distribution de la génération de chaque type de tâche planifiée (Cfr. Ecran 15 page 83).

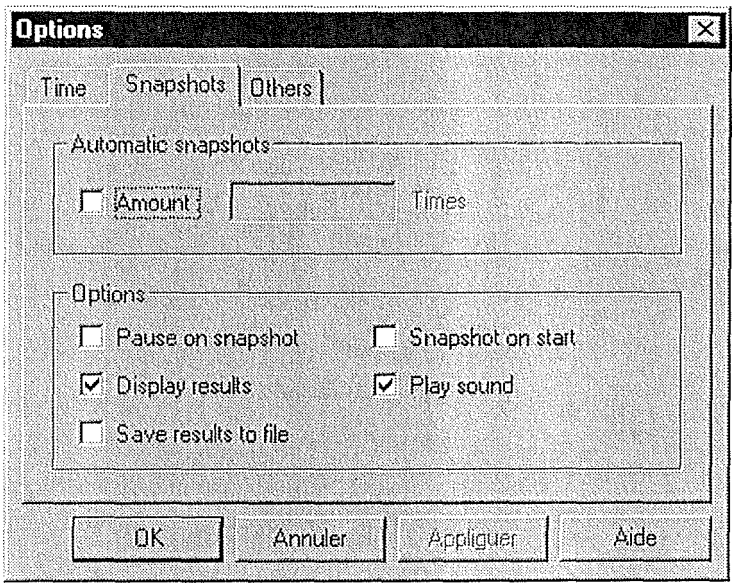
4.6.4 Spécification des paramètres de simulation

Avant de lancer la simulation proprement dite, l'utilisateur peut spécifier différents paramètres, à savoir :

- Le pourcentage du temps fictif de simulation : si la simulation s'étend sur trop de temps, il peut demander de n'en simuler qu'une partie, les différents résultats étant modifiés en conséquence.

- La fréquence des mesures du système (« snapshots ») ainsi que certaines autres options (sauvegarde vers un fichier, avertissement sonore, etc.).
- Le nombre de points dans les graphiques.
- La réaction à simuler lorsqu'une ligne est coupée : recalculer les routes ou laisser faire.

Ces différentes options sont accessibles via la fenêtre d'options présentée ci-dessous (Ecran 18).



Ecran 18 : La fenêtre d'options

4.6.5 Lancement de la simulation

Lorsque le réseau, les tâches et les options de simulation ont été spécifiés, la simulation proprement dite peut enfin commencer. Celle-ci peut être lancée grâce à la barre d'outils ou par l'intermédiaire des menus déroulants.

Modnet effectue d'abord une vérification de l'intégrité du modèle puis lance la simulation si aucune erreur grave n'est rencontrée. La barre de progression est alors affichée afin que l'utilisateur puisse déjà avoir une idée du temps que prendra la simulation.

4.6.6 Affichage de graphiques

Pendant la simulation, en pressant le bouton droit de la souris sur un objet du modèle, l'utilisateur peut à tout moment choisir un noeud ou une ligne et demander l'affichage d'un graphique retraçant l'évolution d'une ou plusieurs variables propres à l'objet sélectionné. Les valeurs affichées sont, pour la plupart, une moyenne depuis la valeur précédente.

Les informations affichables sous forme de graphiques sont :

pour tous les noeuds :

- le nombre tâche « de transit » (qui routent les données d'un noeud à un autre)

pour les instituts :

- le nombre de processeurs utilisés (moyenne depuis le dernier point)
- pour chaque tâche de l'institut :
    - le nombre de tâches générées
    - le nombre de tâches terminées
    - le nombre de tâches en cours d'exécution
    - le temps moyen d'exécution
    - la fréquence de génération

pour les sous-réseaux :

- la quantité de données qui traversent le sous-réseau

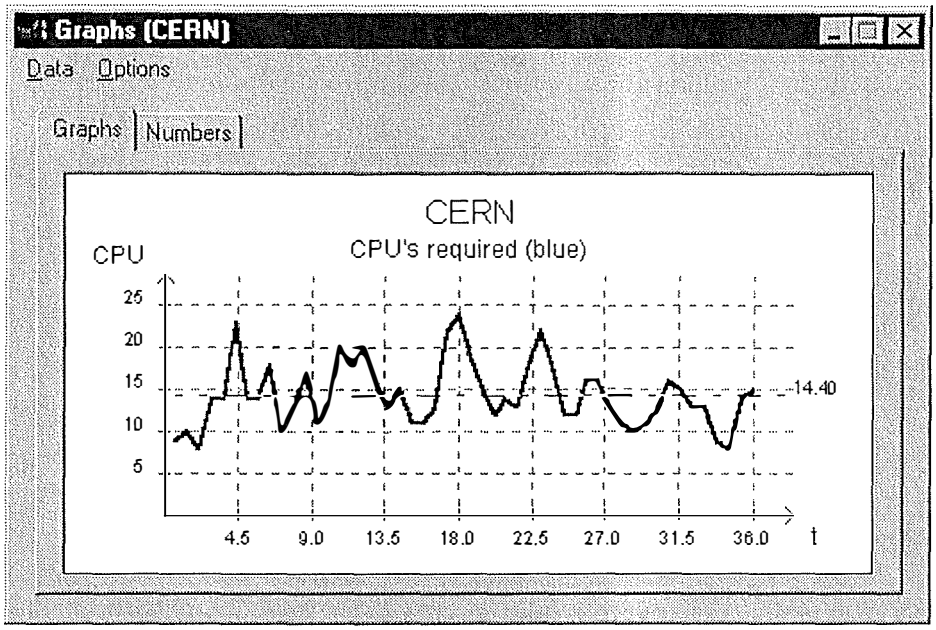
pour les lignes (dans chaque sens) :

- le trafic de base (« background traffic »)
- le trafic de la simulation (généré par les tâches)
- le trafic total (somme des deux premières variables)
- le nombre de canaux ouverts

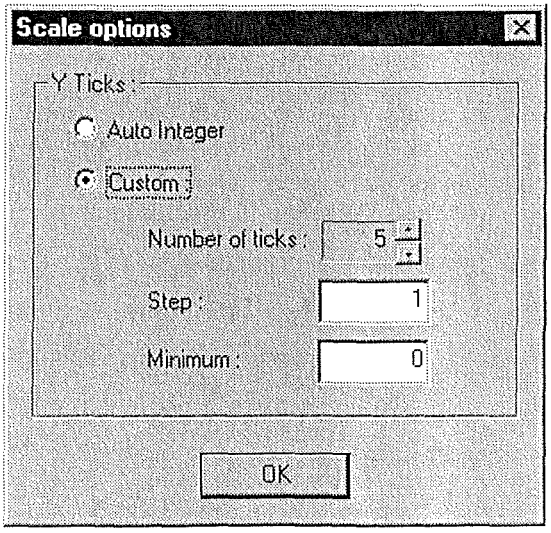
L'échelle utilisée lors de l'affichage des graphiques peut être soit automatique soit définie par l'utilisateur. Dans le premier cas, selon les valeurs affichées, Modnet choisit lui-même l'échelle la mieux adaptée (graduée par des valeurs entières). Dans le second, l'utilisateur peut spécifier sa propre échelle par l'intermédiaire de la fenêtre prévue à cet effet (Ecran 20).

Outre les valeurs elles-mêmes, des données statistiques peuvent être ajoutées au graphique, à savoir : la moyenne, l'évolution de la moyenne ou les limites (minimum et maximum). Un exemple de graphique est également donné à l'Ecran 19.

Notons enfin qu'à tout moment l'utilisateur a la possibilité d'obtenir la liste des valeurs affichées dans les graphiques afin d'avoir une idée précise des résultats présentés. Cette liste est facilement accessible via la page prévue à cet effet dans la fenêtre des graphiques également visible sous le nom « Numbers » dans la fenêtre ci-dessous.



Ecran 19 : Exemple de graphique (nombre de processeurs utilisés + moyenne)



Ecran 20 : La fenêtre de définition d'échelles

4.7 Validation et limitations

4.7.1 Validation

Jusqu'à présent, les seules validations effectuées sur les résultats de Modnet consistent en la vérification de la pertinence de ceux-ci sur des exemples simples. Une fois de plus, il est regrettable de constater que cette démarche importante dans le cas des logiciels de simulation n'a toujours pas été planifiée, comme pour la précédente version du programme. Bien sûr, il serait illusoire d'espérer des résultats comparables à la réalité vu que Modnet fait abstraction, entre autres, des phénomènes aléatoires dus au fonctionnement des couches qui supportent le niveau « application », notamment les niveaux de transport (TCP) et inter-réseaux (IP). Il pourrait cependant être utile de vérifier, sur un réseau de taille moyenne (trois ou quatre institutions européennes, par exemple) si les prévisions de ressources proposées par Modnet sont dans le même ordre de grandeur que les observations faites sur ce réseau et s'il est sensible de la même manière à des changements de paramètres comme la fréquence de génération des tâches.

Si ce genre de validation n'a pas encore été effectué, c'est probablement aussi parce que le projet Modnet n'est encore à la fleur de l'âge. En effet, l'absence de cachier des charges couplée avec le caractère flexible de l'environnement de travail du CERN ont fait germer breaucoup d'idées originales pour améliorer le simulateur. Certaines de ces idées sont présentées au chapitre suivant.

4.7.2 Conclusion

Si l'on ramène Modnet à l'objectif qui lui incombe, ce produit a effectué un grand pas en avant par rapport à son prédécesseur et devrait suffire à la simulation à long terme de réseaux de données. Bien que comportant un grand nombre d'hypothèses restrictives quant au comportement du réseau modélisé, Modnet possède un moteur de simulation plus riche et offre à présent moultes outils pour obtenir des résultats intéressants et exploitables.

5. Critiques et améliorations proposées

Modnet a fait un grand pas en avant dans la simulation de réseaux et les résultats obtenus ont semblé suffire à la communauté scientifique pour soutenir l'une ou l'autre configuration. Néanmoins, si le délai de développement accordé avait été plus long, on aurait certainement pu proposer un modèle différent et des outils supplémentaires...

5.1 Du mode "tranche de temps" au mode événementiel

5.1.1 Le mode "tranche de temps" de Modnet

Modnet est un simulateur basé sur le concept de tranche de temps. Les tranches de temps sont le résultat de la découpe du temps de simulation en intervalles relativement courts (au moins inférieurs au dixième de la durée minimale de chaque tâche) après lesquels une mise à jour complète des différentes variables du modèle est effectuée.

Cette méthode comporte quelques avantages intéressants. Premièrement, elle permet à l'utilisateur de suivre facilement la progression de chaque tâche en observant simplement l'évolution des variables du modèle au cours du temps. L'interface du simulateur se limite donc en une couche de formatage qui présente ces informations à l'utilisateur au fur et à mesure de l'écoulement des tranches de temps.

Ensuite, elle simplifie les procédures de génération et de gestion des tâches. En effet, c'est à la fin de chaque tranche de temps que l'on choisit de générer ou non une nouvelle tâche et de mettre à jour les tâches déjà en cours d'exécution. Bref, aucune planification n'est nécessaire : la tranche de temps est la référence en matière de maintenance du système.

La taille de la tranche de temps est calculée de manière à ce qu'elle soit suffisamment petite pour assurer un minimum de précision dans les résultats. En effet, théoriquement parlant, le temps d'exécution de chaque tâche est entaché d'une erreur d'en moyenne un quart de tranche de temps (cfr. Schéma 18). On comprend dès lors qu'il soit préférable d'en réduire la taille afin d'éviter une trop grande imprécision. Si en pratique les paramètres du modèle sont souvent des multiples de la tranche de temps (et donc annulent l'imprécision ci-dessus), cette approximation est bien réelle et doit être prise en compte dans l'interprétation des résultats.

Le principal inconvénient de cette méthode, dans le cas d'un logiciel comme Modnet, est le temps de simulation. De fait, mettre à jour l'ensemble des variables du modèle après chaque tranche de temps peut devenir une tâche très lourde dès que le nombre de tâches à traiter devient grand. Quand on sait que les simulations prévues feront tourner plusieurs milliers de tâches par minute et que la tranche de temps est inférieure au centième de seconde (vu la rapidité des lignes et la taille réduite des

paquets de données[13]), on imagine bien le nombre de mises à jour nécessaires pour maintenir l'état du modèle et, par conséquent, la durée du temps de simulation.

Par exemple, simuler une heure de transferts de données au sein d'un réseau équipé de lignes à 1 Mbps avec des paquets de données de 64 Ko nécessite plus de 70000 tranches de temps. Le nombre de tâches à gérer dépassant facilement la centaine, les mises à jour à effectuer à la fin de chaque tranche rendent la simulation relativement lourde.

Enfin, conceptuellement parlant, la méthode de la tranche de temps est la moins "belle". D'autres méthodes plus efficaces et plus fines auraient pu être implémentées dans le moteur de simulation de Modnet. Le mode événementiel fait partie de ces méthodes...

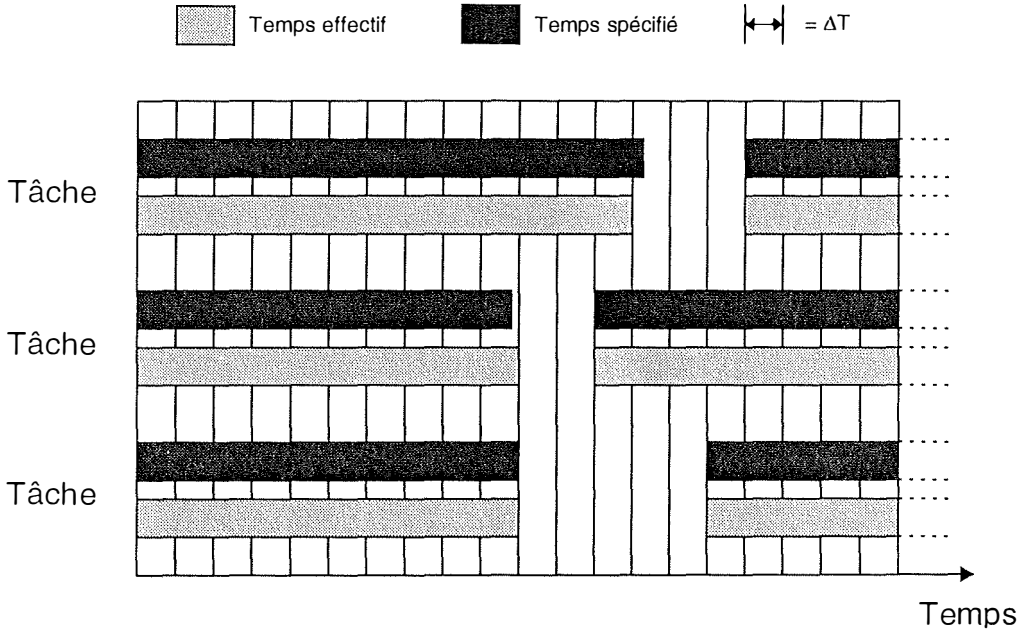


Schéma 18 : les imprécisions dues aux tranches de temps

5.1.2 Le mode événementiel

Comme son nom l'indique, la méthode du mode événementiel se base sur le concept d'événement. Dans le cas de Modnet, la notion d'événement peut être définie comme toute modification de l'état du modèle susceptible d'en modifier la progression. Par exemple, la création d'une nouvelle tâche est un événement dont il faut tenir compte, car des ressources devront être réservées, parfois au détriment d'autres tâches déjà en cours d'exécution.

[13] Voir section 2.2.4. relative au calcul de la tranche de temps.

Une différence fondamentale entre la méthode de la tranche de temps et le mode événementiel est la découpe du temps de simulation. En effet, la première méthode décompose le temps de simulation en intervalles parfaitement réguliers et généralement courts alors que la seconde le découpe en tranches de durée tout-à-fait variable mais relativement longue par rapport à la tranche de temps. Ainsi, grâce au nombre réduit de mises à jour, le mode événementiel permet un gain de temps sensible en comparaison avec la méthode précédente.

Les tranches de temps étant plus longues et de taille variable, la progression des tâches en cours d'exécution n'est a priori pas observable. Celle-ci est donc considérée comme constante tant qu'un autre événement ne vient pas perturber la distribution des ressources utilisées. Par exemple, si lors d'une tâche de transfert entre deux instituts, une autre tâche est générée, la première devra partager la ligne avec la seconde et donc retarder son échéance. (Cfr. exemple ci-dessous)

Le problème central du mode événementiel consiste donc à mettre en évidence tous les événements importants et à identifier les modifications que chacun apporte aux variables du système et à la planification des autres événements. Voici une proposition de liste d'événements remarquables dans la version actuelle de Modnet :

- Création d'une tâche
- Fin d'exécution d'une tâche
- Démarrage de l'étape d'une tâche
- Fin d'exécution de l'étape d'une tâche
- Demande d'une connexion avec un autre noeud du réseau
- Fin d'établissement d'une connexion avec un autre noeud du réseau
- Notification de fin d'une connexion avec un autre noeud du réseau
- Fin de suppression d'une connexion avec un autre noeud du réseau
- Modification du trafic d'arrière-plan sur une ligne
- Modification du nombre de tâches qui utilisent une ligne
- Demande d'affichage des valeurs du système à l'écran

A chaque événement est associée une série de paramètres comme le temps de déclenchement, le type d'événement, l'objet concerné, etc. Le temps de déclenchement doit toujours être calculable lors de la création de l'événement. Bien sûr, ce calcul s'effectue compte tenu du contexte du système au moment de la génération de l'événement et aucun temps de déclenchement n'est à l'abri de modifications au cours de la simulation.

Pour reprendre l'exemple de tâche de transfert, voici un exemple simplifié de déroulement de deux tâches qui se partagent une ligne. La situation du système est résumée du Tableau 6 au Tableau 9 et l'évolution du volume de données transférées lors de l'exécution des deux tâches est présentée au Graphique 6.

Tableau 6 : situation initiale (t = 120 sec)

T = 100 sec	Evénement
120	Démarrage de la tâche T (Transfert de 1 Mo de A vers B)
140	Démarrage de la tâche U (Transfert de 800 Ko de A vers B)

Tableau 7 : Situation au temps 120 (lancement de la tâche T)

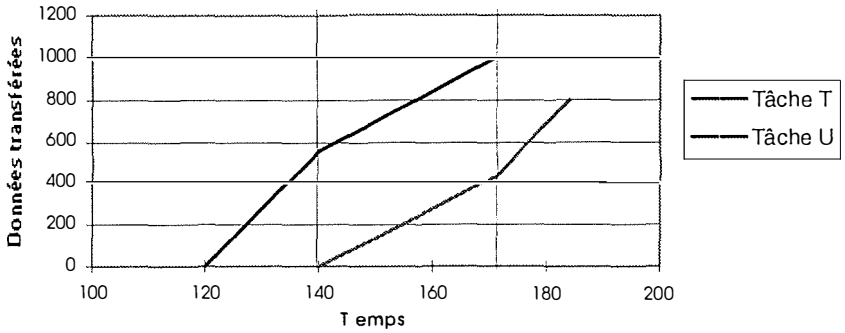
T = 120 sec	Evénement	
140	Démarrage de la tâche U (Transfert de 800 Ko de A vers B)	
156	Fin de l'exécution de la tâche T	
	Type :	Envoi de 1 Mo
	Ligne :	256 Kbps (224 Kbps disponibles)
	Largeur de bande allouée :	224 Kbps
	Progression :	0 Ko
	Durée d'exécution :	36 sec

Tableau 8 : Situation au temps 140 (Lancement de la tâche U)

T = 140 sec	Evénement	
171	Fin de l'exécution de la tâche T	
	Type :	Envoi de 1 Mo
	Ligne :	256 Kbps (224 Kbps disponibles)
	Largeur de bande allouée :	112 Kbps
	Progression :	560 Ko
	Durée d'exécution :	20 + 31 = 51 sec
197	Fin de l'exécution de la tâche U	
	Type :	Envoi de 800 Ko
	Ligne :	256 Kbps (224 Kbps disponibles)
	Largeur de bande allouée :	112 Kbps
	Progression :	0 Ko
	Durée d'exécution :	57 sec

Tableau 9 : Situation au temps 171 (Fin de la tâche T)

T = 171 sec	Evénement	
184	Fin de l'exécution de la tâche U	
	Type :	Envoi de 800 Ko
	Ligne :	256 Kbps (224 Kbps disponibles)
	Largeur de bande allouée :	224 Kbps
	Progression :	434 Ko
	Durée d'exécution :	31 + 13 = 44 sec



Graphique 6 : Evolution des données transférées sur une ligne commune

En conclusion, intégrer le mode événementiel à Modnet semble être une amélioration de choix qui devrait réduire considérablement le temps de simulation et clarifier la relation de cause à effet entre les différentes variables et événements du simulateur. Le Tableau 10 tente d'effectuer une comparaison des deux méthodes selon quelques critères.

Tableau 10 : Comparaison entre le mode "Tranche de temps" et le mode événementiel

	Mode "tranche de temps"	Mode événementiel
Taille de l'intervalle de temps → Durée : → Régularité :	Très petite Taille fixe	Grande Taille variable
Erreur théorique de simulation du temps d'exécution d'une tâche :	En moyenne le quart de la tranche de temps	Inexistante
Nombre de mises à jour :	Très élevé	Réduit
Temps de simulation → Durée : → Paramètre déterminant :	Généralement longue Taille de l'intervalle de temps	Réduite Nombre de tâches à générer
Planification :	Inexistante	Indispensable
Progression des tâches :	Observable	Non observable

5.2 Hiérarchies de réseaux

5.2.1 Situation actuelle et besoins

L'objectif de Modnet est de permettre à l'utilisateur de pouvoir modéliser un réseau s'étendant sur plusieurs continents. Ceci implique souvent des configurations qui regroupent un grand nombre de noeuds et de lignes, ce qui mène vite le modèle vers un état confus et difficile à lire.

L'idéal serait de pouvoir regrouper certaines parties du réseau en des sous-réseaux logiques de manière à apporter une dimension verticale au modèle. Cela permettrait une réduction de la complexité et une amélioration de la lisibilité.

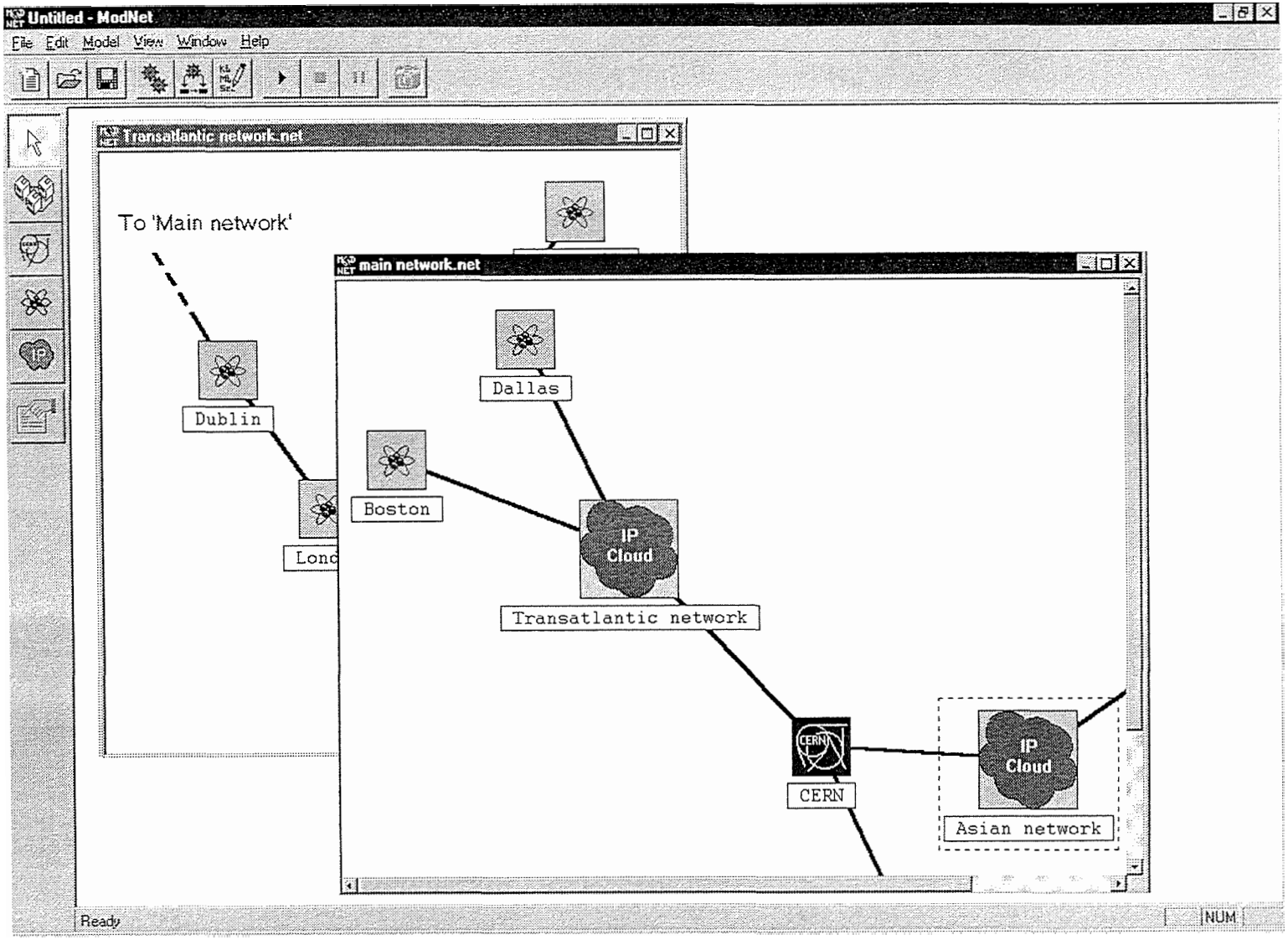
5.2.2 Structuration en hiérarchies

Face au besoin de restructuration, la notion de réseau se prête facilement à la décomposition en sous-réseaux logiques qui deviennent eux-mêmes un noeud du réseau duquel ils faisaient partie. De plus, cette décomposition reste valable autant de niveaux que nécessaire, jusqu'à ce que l'utilisateur soit arrivé à un niveau de décomposition satisfaisant.

Par exemple, on pourrait imaginer un réseau décomposé de manière suivante : le réseau initial comporte de nombreux noeuds et lignes répartis en Europe, aux Etats-Unis et en Asie (principalement en Inde et au Japon). Une première décomposition en sous-réseaux continentaux s'avère être un bon départ. Ensuite, on peut regrouper les noeuds selon un regroupement géographique (par pays) ou encore par domaine d'activité pour terminer par un éventuel troisième niveau si certains noeuds, par exemple, travaillent en étroite collaboration.

5.2.3 Implémentation

Dans un environnement tel que celui de Windows 95, le meilleur moyen d'implémenter ce genre de découpe est d'attribuer une fenêtre par réseau ou sous-réseau et de définir des liens entre chaque fenêtre.



Ecran 3 : Prototype d'une interface MDI de Modnet

La version actuelle de Modnet possède une interface de type SDI (*Single Document Interface*) c'est-à-dire qui ne peut ouvrir qu'un seul document à la fois. Afin de pouvoir bénéficier du multi-fenêtrage, l'interface de l'application doit être revue et devenir de type MDI (*Multiple Document Interface*). L'application étant entièrement de type "orienté objet", les modifications à effectuer ne touchent pas véritablement le coeur du programme. Le principal changement consiste en effet à remplacer le type de l'objet duquel est dérivé la fenêtre principale de Modnet (de type SDI) par un autre qui offre des services MDI de multi-fenêtrage.

Ainsi, chaque modèle sera muni d'un seul document contenant la description complète du réseau et des tâches qui y sont planifiées. Avec ce document pourront être créées plusieurs vues différentes dans des fenêtres séparées, chaque vue pouvant être un (sous-)réseau. (Note: voir chapitre sur l'architecture document-vue).

L'Ecran 21 montre un aperçu de ce que pourrait être l'interface d'une version hiérarchisée de Modnet.

5.3 Simulation de pannes

5.3.1 Situation actuelle et besoins

Modnet permet à l'utilisateur de désactiver certaines lignes avant ou pendant la simulation pour générer une pseudo-panne ou tout autre phénomène empêchant les données d'être transférées. Dans l'état actuel des choses, seule une désactivation manuelle peut-être réalisée et aucune automatisation ne peut être effectuée.

Une possibilité d'amélioration pour Modnet serait donc de permettre à l'utilisateur de spécifier, pour chaque ligne, des paramètres de simulation de pannes tout en lui laissant la possibilité d'intervenir manuellement sur l'état des lignes, comme c'est le cas dans la version actuelle de Modnet.

5.3.2 Paramètres de simulation

Les différents paramètres pouvant intervenir dans la simulation des pannes de lignes sont, pour chaque ligne :

- L'état initial (activée / désactivée)
- La distribution de la fréquence des pannes
- La distribution de la durée des pannes

L'état initial est simplement le caractère "activée" ou "désactivée" donné par l'utilisateur avant le lancement de la simulation. Cette propriété peut déjà être spécifiée dans Modnet grâce au menu contextuel des lignes.

La distribution de la fréquence des pannes pourrait être, entre autres, uniforme, normale ou de Poisson dont la moyenne serait donnée par l'utilisateur.

La distribution de la durée des pannes pourrait, quant à elle, être uniforme, normale ou exponentielle.

L'utilisateur pourrait spécifier ces différents paramètres grâce à une page supplémentaire ajoutée à la fenêtre de propriétés des lignes, une fenêtre de sélection de distribution statistique étant déjà implémentée.

5.3.3 Implémentation

Si le mode de la tranche de temps est maintenu, la fréquence des pannes peut être ramenée à une fréquence par tranche de temps à tester continuellement. Une autre solution serait de découper le temps inter-panne en tranches de temps, avec l'imprécision qui s'ensuit. Cette dernière solution est également applicable à la simulation de la durée des pannes.

Si le mode événementiel est implémenté, la survenance et la fin d'une panne deviennent deux nouveaux événements du système. La fin d'une panne est programmée dès le traitement du début de la panne. Quand la fin de panne est traitée, la survenance suivante est programmée etc.

6. Conclusion

6.1 Les besoins du CERN

Le CERN avait besoin d'un outil simple, sur mesure, sans trop de paramètres.

Modnet fait abstraction du type de matériel utilisé et du protocole réseau ou inter-réseaux mis en place. Les paramètres sont réduits au strict minimum et ne requièrent pas d'analyse préliminaire quant aux moyens techniques utilisés lors de la mise en service du réseau. Par conséquent, Modnet n'a pas la prétention de fournir une information décisive dans la prise de décision, mais plutôt un avis a priori.

Le CERN avait besoin d'un outil muni d'une interface riche et conviviale.

Basé sur l'architecture MFC de Microsoft (c) , Modnet possède une interface intuitive et facile d'utilisation. Les barres d'outils, les menus déroulants, les fenêtres de propriétés et les menus contextuels sont autant d'avantages qui permettent à Modnet d'être à la hauteur des espérances des utilisateurs.

Le CERN avait besoin d'une architecture orientée objet dans un souci de réutilisabilité.

Chaque concept manipulé par Modnet est un objet redéfinissable et réutilisable. L'ajout simple de nouveaux types de tâches ou de noeuds a été un critère important lors de la conception de l'architecture.

Le CERN avait besoin d'un modèle plus riche et plus précis.

Modnet intègre à présent les notions de sous-réseau, de background traffic, de distribution statistique, d'unité utilisateur, de coût de signalisation, de route, de puissance de calcul variable, etc. Tous ces nouveaux concepts apportent des informations supplémentaires aux résultats de la simulation et ajoutent une touche de réalisme au modèle. Modnet conserve néanmoins la méthode de la découpe en tranches de temps qu'il serait intéressant de remplacer par un système événementiel afin de garantir une simulation plus rapide et plus précise.

6.2 Le problème de la validation

Tout simulateur, afin que les résultats qu'il donne fassent partie d'un processus de décision, doit être passé par un processus de validation qui permette d'identifier dans quelle mesure les résultats fournis correspondent à la réalité.

Dans le cas de Modnet, comme dans celui du simulateur précédent, aucune validation proprement dite n'a été réalisée. D'ailleurs, le nombre restreint de paramètres et la quantité d'hypothèses restrictives rendraient toute comparaison plutôt aventureuse quand on connaît la complexité d'un réseau WAN. On pourrait donc craindre une certaine tiédeur de la part des décideurs vis-à-vis de ces simulateurs.

6.3 Perspectives

Le projet "Modnet" a apporté un second souffle au simulateur existant et a permis au chef de projet de diffuser davantage son produit parmi la population des utilisateurs potentiels grâce à une meilleure interface.

Dans l'état actuel des choses, Modnet n'est pas encore distribué et utilisé; certaines améliorations peuvent encore être faites avant de pouvoir en tirer un maximum d'informations. C'est probablement à cause de l'enthousiasme engendré par cette nouvelle version ainsi que l'absence totale de cahier des charges avant ou pendant son développement que l'équipe du projet a décidé, à juste titre, d'aller plus loin.

Annexes

Annexe 1 : Les tableaux de la première version du simulateur.

Annexe 2 : Présentation du simulateur à la commission CMS du CERN.

Annexe 3 : L'algorithme de Moore-Dijkstra et sa démonstration.

Annexe 1 : Les tableaux de la première version du simulateur

A1.1 La taille des tableaux

La taille des tableaux est fixée par une série de constantes définies par le programmeur ; en cas de changement de valeur, le programme doit donc être recompilé. Le tableau 5 ci-dessous reprend la liste de ces constantes.

Tableau 11 : Les constantes de limitation des tableaux

Nom	Signification	Valeur par défaut
maxinst	Nombre maximum d'instituts	200
maxvc	Nombre maximum de connexions ouvertes lors de la simulation	100 000
maxtask	Nombre maximum de tâches en cours d'exécution	100 000
maxsteps	Nombre maximum d'étapes dans une tâche	6
maxtypes	Nombre maximum de types de tâches différents dans le modèle	10
maxmix	Nombre maximum de types de tâches planifiés par un institut	10

A1.2 Les tableaux du programme

Tableau 6 : Les différents tableaux du programme

Nom	Information stockée	Dimension
source	Numéro de l'institut qui envoie les données sur la connexion	maxvc
sink	Numéro de l'institut qui reçoit les données sur la connexion	maxvc
depend	Numéro de l'institut qui a demandé l'ouverture de la connexion	maxvc
rate	largeur de bande allouée à la connexion (octets / sec)	maxvc
total	Quantité de données à transférer sur la connexion (octets)	maxvc
done	Quantité de données transférées sur la connexion (octets)	maxvc
startvc	Moment auquel a été établie la connexion (sec)	maxvc
location	Numéro de l'institut qui a généré une tâche	maxtask
taskmix	Numéro, dans l'institut, du type de tâche planifié associé	maxtask
taskstep	Numéro de l'étape en cours d'exécution d'une tâche générée	maxtask
tasktype	Numéro de type de tâche d'une tâche générée	maxtask
starttask	Moment auquel a été générée une tâche (sec)	maxtask
startcpu	Moment auquel a commencé une étape CPU d'une tâche (sec)	maxtask

institute	Nom d'un institut	maxinst	
nmix	Nombre de types de tâche planifiés à un institut	maxinst	
ntasks	Nombre de tâches en cours d'exécution à un institut	maxinst	
nincpu	Nombre de tâches en cours de traitement CPU à un institut	maxinst	
taskmax	Nombre maximum de tâches en cours observé à un institut	maxinst	
cpumax	Nombre maximum de tâches en cours de traitement CPU observé à un institut	maxinst	
wanmax	Nombre maximum observé de connexion partant d'un institut	maxinst	
diskused	Espace disque utilisé (octets)	maxinst	
costdisk	coût de l'espace disque ($ / GB)	maxinst	
costnet	coût de location d'une ligne ($ / sex)	maxinst	
costproc	coût d'acquisition d'un processeur 100 MIPS ($)	maxinst	
rateLAN	taux de transfert observé sur le LAN d'un institut (octets / sec)	maxinst	
mix	Numéro de type de tâche pour un type de tâche planifié	maxinst maxmix	x
frequency	Fréquence requise lors de la génération des tâches (sec^{-1})	maxinst maxmix	x
number	Nombre requis de tâches à générer	maxinst maxmix	x
taskstarted	Nombre de tâche générées pour un type de tâche planifié	maxinst maxmix	x
taskfinishd	Nombre de tâches terminées pour un type de tâche planifié	maxinst maxmix	x
tasktime	Temps moyen d'exécution des tâches d'un type de tâche (s)	maxinst maxmix	x
connect	Nombre de connexions en cours entre deux instituts	maxinst x maxinst	
ratematrix	Largeur de bande d'une ligne entre 2 instituts (MB / sec)	maxinst x maxinst	
taskname	Nom d'un type de tâche	maxtypes	
steps	Nombre d'étapes d'un type de tâche	maxtypes	
steptype	Type d'une étape d'un type de tâche	maxtypes maxsteps	x
stepsize	Taille d'une étape d'un type de tâche (octets ou sec)	maxtypes maxsteps	x

Annexe 2 : Présentation du simulateur à la commission CMS du CERN par Julian BUNN [BUNN97]

A2.1. Introduction

We describe, and show results from, a simple program for simulating the various LHC Computing Models. Currently, the models of particular interest include:

1) Regional Centres Model
2) Fully Distributed Model
3) Ce(r)ntric Model

The simulation tool allows us to identify, for each model, probable required bandwidths between the collaborating institutes, limits on the types and number of computing tasks that could be carried out at the collaborating institutes, and order-of-magnitude estimates of the financial cost of the required computing infrastructure.

A2.2. Background

The simulation technique is very loosely based on work done for SHIFT in the early 90's. Once enough experience has been gained with this simulation, a more sophisticated simulation should be constructed using a commercial WAN modelling tool such as COMNET III. Initial impressions of COMNET III as a tool for this purpose are included at the end of this document.

A2.3. Simulation Specification

The simulation program allows for model features that include:

1) Any number of collaborating institutes
2) Several computing task types (Reconstruction, Analysis, Detector Studies, etc..)
3) Bandwidth limited inter-institute connections

The simulation program calculates the following information:

A) Aggregate and instantaneous WAN and LAN data rates for each institute
B) Turnround times for each type of task at each institute
C) Approximate model costs based on commodity hardware

A2.4. Model Parameters

The following entities are specified as model input:

→ The collaborating institutes, each institute being characterised by the following data (specified by the modeller):

1) Institute Name (e.g. "CERN", "FNAL", etc..)
2) WAN bandwidth to each other institute
3) LAN bandwidth
4) Task mix and associated "probabilities" (see below)
5) Infrastructure unit costs

→ The computing tasks that may be run at the collaborating institutes, characterised by the following data (specified by the modeller):

1) Task Name (e.g. "Analysis", "Reconstruction", "Detector Studies", etc..)
2) Number of "steps" in the task
3) The type of each step (e.g. "Get data", "Put data", "Process data", etc..)
4) The "size" of the step (e.g. number of MBytes for "Get data", or number of CPU seconds on a 100 MIPS machine for "Process data")

A2.5. Simulation Method

The simulation is based on time slices. The clock starts at time T=0.0 and ticks every delta seconds, where delta is chosen to give 10 clock ticks for the smallest data set transfer at the largest available data rate. This ensures sufficiently accurate simulation of small data set transfers.

At each clock tick, the task mix for each collaborating institute is examined. One or more tasks may then be created at the institute, depending on the assigned "probability" of the task when compared with the value of a random number. Example: At FNAL, a "Reconstruction" task of "probability" 100.0 will be started, on average, every 100 seconds. Alternatively, if the task mix assigns a total number to a task, then the required frequency to achieve this number is assigned to the task.

Each created task begins at step 1 of its sequence. A "reconstruction" task might be specified as follows:

```
Nsteps = 3
   Step 1 : Get 1.0 MBytes of Raw Data from CERN
   Step 2 : Process for 100 elapsed seconds
   Step 3 : Put 50 kBytes of Reconstructed Data to CERN
```

whereas a "simulation" task might be specified as:

```
Nsteps = 2
      Step 1 : Process for 100 elapsed seconds
      Step 2 : Put 1.0 MBytes of Simulated data to "local"
file
```

At each clock tick, every task is examined to determine whether the current step of the task has completed. If so, then the task is either moved on to its next step, or marked as having terminated.

For tasks where a total number is to be achieved, the assigned frequency for the task is re-computed at each time step accordingly.

The simulation continues running until the time limit (specified by the modeller) is reached, whereupon various statistics are printed out.

A2.6. Virtual Circuits: Simulation of the network

For tasks that begin a network transfer step, a "virtual circuit" is established between the institute at which the task is running, and the institute where the data reside. For connections between two different institutes, an Adjustable Bit Rate (ABR) mechanism is assumed for the ATM switch. Each new virtual circuit is thus assigned a data rate for the transfer calculated as the WAN bandwidth model parameter divided by the number of currently active virtual circuits between the two institutes.

This calculation is re-done whenever a virtual circuit is created or destroyed on the switch.

For local connections, the virtual circuit is assigned the full ATM rate, and an aggregate rate is calculated and histogrammed for the LAN switch at each time step. The intention here is to allow potentially unlimited backplane or site-wide LAN rates.

The histogram of the aggregate rates that actually occured during the simulation gives an indication of the required speed of access to storage implied by the rest of the model.

A2.7. Costing the models

To obtain a very approximate idea of the financial cost implied by a particular computing model, we specify the following parameters for each institute:

A = the cost in dollars of a 100 MIPS processor
B = the cost in dollars of 1 GByte of disk space
C = the cost in dollars of a 1 month lease of a 2 Mbits/sec network connection to a remote institute

(For dollars read "arbitrary currency unit" in the list above.) Then, at the end of the simulation, we calculate for each institute :

1) the cost of the required processing power (i.e. the peak number of 100 MIPS processors that was required during the simulation multiplied by A),
2) the cost of the disk capacity (i.e. the total disk capacity required to store both data read, and data written, multiplied by B), and
3) the cost of the WAN links (i.e. the number of WAN links required, multiplied by the ATM rate as a fraction of 2 Mbits/sec, multiplied by C).

Most of these costs are of course highly speculative. For example, it is not possible today to lease lines running at more than 2 Mbits/sec in Europe. The tariffs for 2 MBits/sec lines runs at around 30000 CHF for one month of lease of a half-circuit to a country neighbouring Switzerland. We thus use a sliding scale that gives a cost of 30000/1.5 = 20000 dollars for a 2 Mbits/sec line, and double that for a 4 Mbits/sec line, and so on...

In the end we sum all the institute costs to arrive at a global model cost figure. The global cost represents the cost of the infrastructure required to perform the given task. The processor part of the cost is invariant under the number of times the given task needs to be performed because, given that the specified processing capacity is installed, then those processors can be used for a continuous series of such tasks. The network costs, on the other hand, go in multiples of a month. If reconstruction

continues for 100 days, then three months of lease must be paid, and the network cost must be multiplied by three. The cost of the disk space required for such a run depends on whether the disks are used as a cache or as a permanent store of the data. This area requires further study and modelling.

A2.8. Implementation

The prototype simulation program is coded in Fortran 77 and works by first reading in a description file that contains the parameters for the required run.

The production tool has been completely re-written in Visual C++ using MFC (Microsoft Foundation Classes), and has a GUI that allows full control over all elements of the model, including real-time graphing of traffic and load. The screen shot below shows the production tool in use.[14]

A2.9. Simulation Results

We show in the following sections results from simulating the various scenarios. These results are also tabulated in an Excel spreadsheet. Note that, since ATM offers only about a 70% payload factor (i.e. the actual data that can be moved across an ATM switch is equivalent to about 70% of the nominal switch rate), the network costs shown in the figures below are underestimated by that amount.

☞ *One Day Full Analysis of 1% of the data*

Using the program described, we have simulated the task of a full analysis of a single physics channel in the Central, Regional Centres and Fully Distributed scenarios, under the following assumptions:

1. Each institute has 20% of the reconstructed data available locally.
2. The remaining reconstructed data is held at CERN.
3. Each Analysis requires to process 1% of the total data (10^7 events).
4. The Analysis time per event on a 100 MIPS processor is 1 second.
5. The Analysis is performed in a production environment: i.e. a full pass over all data in one day.

[14] Cet copie d'écran est disponible à la section 4.4.1. relative à l'interface du logiciel.

Note that the Analysis requires there to already be available 10^7 simulated events: the generation of these events is not modelled.

In the table below, we show the WAN data rates at CERN implied by the task size. For comparison, the average rate seen leaving CERN on WAN lines is currently around 0.5 MBytes/sec (i.e. 40 GBytes per day).

Configuration	Data Moved to Institutes (GBytes)	Aggregate Rate at CERN (MBytes/sec)
Central	0	0
Regional	350	4.0
Distributed	398	4.6

We show in the figure below the cost variations in the three cases Central analysis, Regional analysis and Distributed analysis. The infrastructure costs used were:

- 200$ per GByte of disk space
- 20000$ per 2 Mbits/sec leased line (i.e. 40000$ for 4 Mbits/sec etc.)
- 2000$ for a 100 MIPS processor

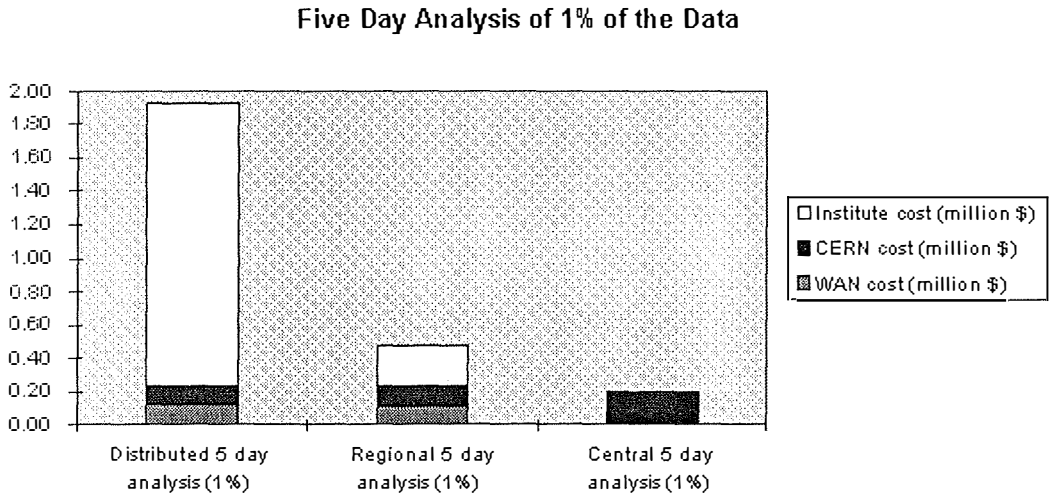
The WAN costs are calculated using the minimum possible speed required to complete the analysis in the alloted time.

☞ *Five Day Full Analysis of 1% of the data*

Under the same conditions, we describe an analysis of 1% of the data in a run over 5 days.

We show again the WAN data rates at CERN implied by the task size.

Configuration	Data Moved to Institutes (GBytes)	Aggregate Rate at CERN (MBytes/sec)
Central	0	0
Regional	350	0.8
Distributed	398	0.9

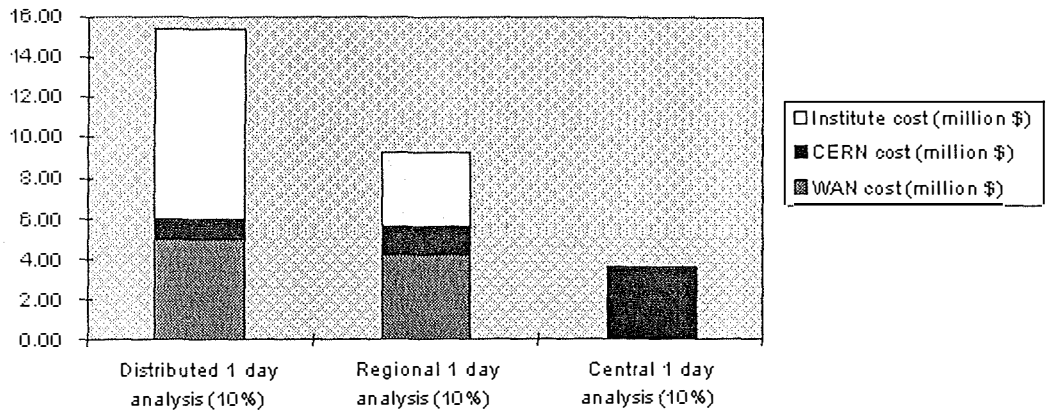


☞ *One Day Full Analysis of 10% of the data*

Under the same conditions, we describe an analysis of 10% of the data in a run over a single day.

We show again the WAN data rates at CERN implied by the task size.

Configuration	Data Moved to Institutes (GBytes)	Aggregate Rate at CERN (MBytes/sec)
Central	0	0
Regional	3500	40.0
Distributed	3980	46



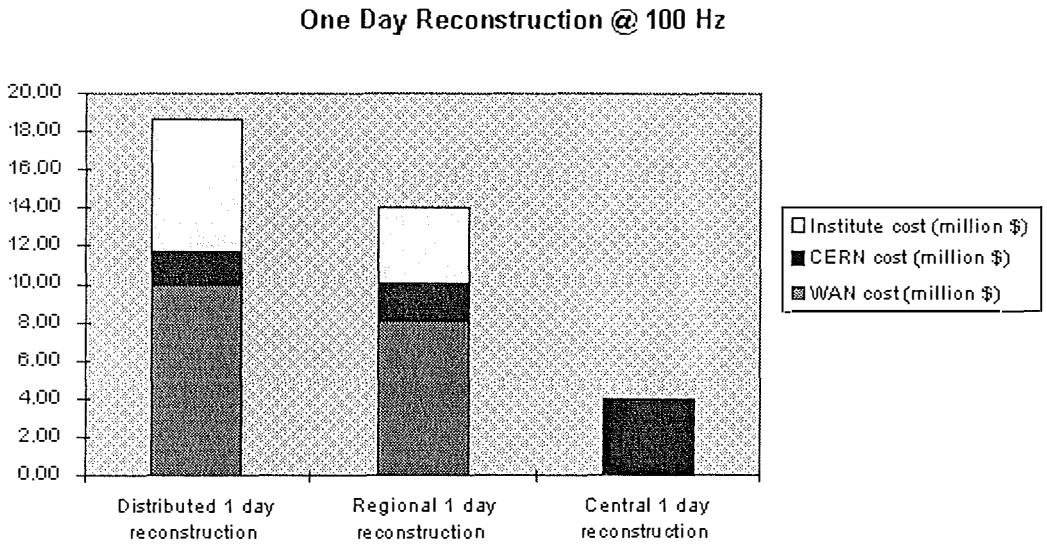
☞ *Synchronous 100Hz Reconstruction*

Using the program, we have also simulated the task of synchronous LHC event reconstruction in the Central, Regional Centres, and Fully Distributed scenarios, under the following assumptions:

1. A Trigger Level III raw event rate of 100 Hz.
2. A raw event size of 1 MByte.
3. A reconstruction time of 10 seconds on a 100 MIPS processor.
4. Raw events stored centrally at CERN.

Here are the WAN data rates at CERN implied by the task size.

Configuration	Data Moved to Institutes (GBytes)	Aggregate Rate at CERN (MBytes/sec)
Central	0	0
Regional	7560	87.5
Distributed	8597	99.5



Appendix: Preliminary experience with COMNET III for the LHC Computing Modelling

As a learning exercise with the commercial WAN modelling tool COMNET III, we have made a very simple LHC computing model that can be seen in the figure below[15].

The model is simple in that it simulates only one remote institute. The remote institute is connected to CERN via a WAN ATM cloud via CISCO-type routers at each end. At the remote institute are modelled three different workstation types: one that is running simulation, one that is running analysis, and one that is running reconstruction. Bear in mind that all components of the model as shown can be replicated and grouped as necessary. This means that, once one is satisfied that the components of the simple model are being simulated correctly, then one can scale up the simulation by, for example, having twenty simulation workstations, and fifty remote institutes.

Returning to the model, the tasks that run on the workstations are modelled as step-wise sequencies. The "Reconstruct Event Remote" task, for example, consists of the following atomic steps:

- RequestRawEvent - This causes a message to be sent across the network to CERN, requesting the next Raw Event.
- GotRawEvent - This is the Raw Event data itself, and is special form of trigger message.
- ReconstructOneEvent - This step simulates computation on the host processor for a time dependent on the Raw Event data size.
- WriteOneReconstructedEventLocal - This steps simulates writing the Reconstructed Event data to a local disk.

The task itself can be scheduled to run in many different ways. These include the task running every so many seconds, the task running for so many events, the task running randomly according to a user-defined probability function, or the task running on reception of a message. In the model shown I chose the task to run to reconstruct 1000 events i.e. in a loop.

[15] Cette copie d'écran est disponible

The model, when run, can be animated and real-time graphs can be plotted of, for example, congestion on the CERN FDDI, or I/O rates to the local workstation disks. This is quite good fun to watch. In fact, building a model using this tool is entertaining, but there is a rather steep initial learning curve. The development of various "scenarios" is well supported, allowing the modeller to see the effects of, for example, a network link becoming unavailable, or the effects of an disk I/O bottleneck at CERN.

Julian J. Bunn

Annexe 3 : L'algorithme de Moore-Dijkstra et sa démonstration.

A3.1 L'algorithme

L'algorithme présenté ci-dessous [FICH95] est la version utilisée lorsque le poids des arcs est égal à 1.

Initialisation

$D \leftarrow \{x_r\}$
$\overline{D} \leftarrow X / \{x_r\}$
$dist(x_r) \leftarrow 0$
$\forall x \in \overline{D}$,
    $dist(x) \leftarrow +\infty$

Boucle principale

tant que $D \neq G$,
    choisir $y \in \overline{D}$ tel que $dist(y) = \min dist(x) \mid x \in \overline{D}$
    $D \leftarrow D \cup \{y\}$
    $\forall x \in \overline{D}$,
        $dist(x) \leftarrow \min (dist(x), dist(y) + poids\_arc(y, x))$

A3.2 Le théorème de base

Afin de montrer que l'algorithme de Dijkstra est correct, il faut supposer que les étiquettes *dist(x)* sur les arcs ne sont pas négatives et que le graphe ne comporte pas de boucles (cfr. Propriétés P1 et P2, page 78). La démonstration [ULLM93] tentera de prouver, par récurrence sur *k*, que lorsqu'il y a *k* sommets fixés,

(a) Pour chaque sommet fixé *u*, *dist(u)* est la distance minimale de *r* à *u*, et le plus court chemin jusqu'à *u* contient seulement des sommets fixés.
(b) Pour chaque sommet non fixé *u*, *dist(u)* est la distance minimale de tout chemin spécial[16] de *s* à *u* ($+\infty$ si un tel chemin n'existe pas).

[16] Un *chemin spécial* étant un chemin composé de sommets appartenant à D puis de sommets appartenant à $\overline{D}$

LA BASE. Pour $k = 1$, s est le seul sommet fixé. On initialise *dist(s)* par 0, satisfaisant (a). Pour tout autre sommet u, on initialise *dist(u)* par l'étiquette de l'arc $s \rightarrow u$ s'il existe, et $+\infty$ sinon. Donc (b) est satisfait.

LA RECURRENCE. Maintenant, si l'on admet que (a) et (b) sont respectés, même après que k sommets ont été fixés, et soit v, le $(k+1)$ sommet fixé.

On affirme que (a) tient encore, car *dist(v)* est la plus petite distance pour tout chemin de s à v. En effet, s'il on suppose le contraire, d'après la partie (b) de l'hypothèse de base, lorsque k sommets sont fixés, *dist(v)* est la distance minimale de n'importe que chemin spécial vers v ; et donc, il doit y avoir un chemin non spécial plus court vers v. Comme indiqué dans le schéma ci-dessous, le chemin doit passer par les sommets fixés à un sommet w (qui peut être s) et aller vers un sommet non fixé u. A partir de là, le chemin peut faire des méandres à partir et vers des sommets fixés, jusqu'à ce qu'il arrive à v.

Cependant, v a été choisi pour être le $(k+1)^{ème}$ sommet fixé, à ce moment, la *dist(u)* ne devrait pas être inférieure à *dist(v)*, ou sinon il aurait fallu sélectionner u comme le $(k+1)^{ème}$ sommet. D'après l'hypothèse de base (b), *dist(u)* est la longueur minimale de tout chemin spécial vers u. Mais le chemin de s à w puis à u dans le schéma est un chemin spécial, tel que sa distance est au moins *dist(u)*. Donc, le supposé chemin le plus court de s à v passant par w et u a une distance qui est au moins *dist(v)*, car la partie initiale de s à u est déjà distante de *dist(u)*, et $dist(u) \geq dist(v)$. Donc, (a) tient toujours pour $k+1$ sommets, c'est-à-dire, (a) est maintenue lorsque l'on inclut v parmi les sommets fixés.

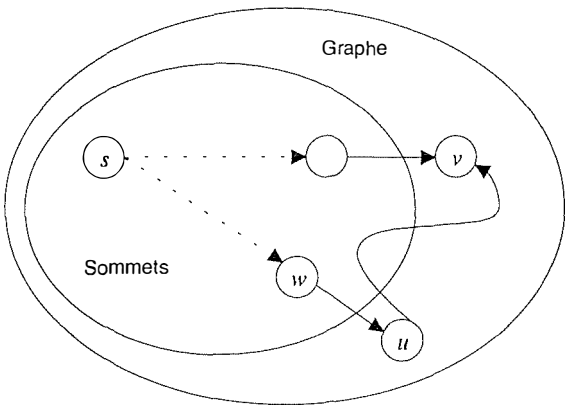


Schéma A1 : Chemin hypothétique le plus court de *v*, passant par *w* et *u*.

Maintenant, il faut montrer que (b) est encore respecté lorsque l'on ajoute v aux sommets fixés. Soit un sommet quelconque u qui reste non-fixé lorsque l'on ajoute v aux sommets fixés. Dans le chemin spécial le plus court vers u, il doit y avoir un pénultième sommet (juste avant le dernier) ; ce sommet devra être soit v, soit un autre sommet w. Les deux possibilités sont indiquées dans le schéma.

D'abord, si l'on suppose que le pénultième noeud est v, la longueur du chemin de s à v puis à u, indique dans le schéma, est *dist(v)* plus l'étiquette de l'arc $v \rightarrow u$.

D'autre part, si l'on suppose que le pénultième sommet est un autre sommet w, d'après l'hypothèse de base (a), le chemin le plus court de s à w est seulement composé de sommets qui étaient fixés antérieurement à v, et par conséquent, v n'apparaît pas dans le chemin. Donc, la longueur du chemin spécial le plus court jusqu'à u ne change pas lorsque l'ont rajoute v aux sommets fixés.

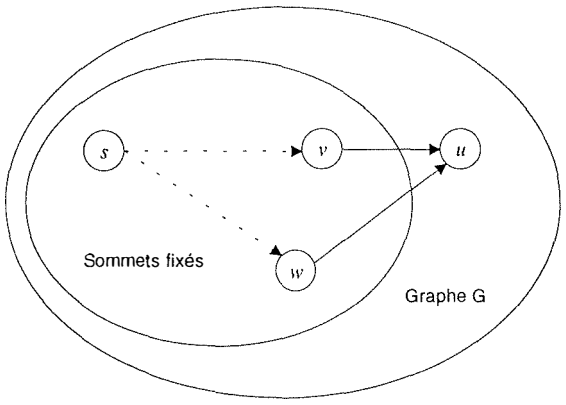


Schéma A2 : Quel est le pénultième sommet du chemin spécial le plus court vers u ?

Maintenant, il est important de rappeler que lorsque v est fixé, on ajuste chaque *dist(u)*pour qu'elle soit la plus petite de l'ancienne valeur *dist(u)* et *dist(v)* plus l'étiquette de l'arc $v \rightarrow u$. Le premier couvre le cas où un w autre que v est le pénultième sommet, et le dernier couvre le cas où v est pénultième sommet. Donc, la partie (b) tient aussi, et la récurrence est achevée.

Table des illustrations

Bibliographie

[FICH95] J. Fichefet, « *Théorie des graphes, notes de cours* », Facultés Universitaires Notre-Dame de la Paix, 1995.

[FERR94] Fr. Ferrier, « *Borland C++ 4.0, programmation avancée* », Cybex, Paris, 1994.

[NOIR97] M. Noirhomme, « *Méthodes de simulation, notes de cours* », Facultés Universitaires Notre-Dame de la Paix, 1997.

[VAND95] J. Vanderdonckt, « *Guide ergonomique des interfaces homme-machine* », Facultés Universitaires Notre-Dame de la Paix, 1995.

[BUNN97] Julian Bunn, « *LHC Computing Models Simulation* », disponible sur le WEB à http ://axcn01.cern.ch/ht_root/model.html, mai 1997.

[CRN97-1] LHC project Webmaster, « *History of LHC* », disponible sur le WEB à http ://www.cern.ch/CERN/LHC/pgs/general/history.html, mai 1997.

[CRN97-2] LHC project Webmaster, « *Cryogenics for the LHC* », disponible sur le WEB à http ://www.cern.ch/CERN/LHC/pgs/general/cryogenics.html, mai 1997.

[ULLM93] Alfred Aho, Jeffrey Ullman, « Concepts fondamentaux de l'informatique », pages 547 à 552, Dunod, Paris, 1993